Document :	*Méta-IHM et points de contrôle utilisateur* dans CONTINUUM
Sous-tâches :	4.1 et 4.2
Numéro des Livrables :	D4.1 et D4.2
Date	06/09/2010
Rédacteurs :	*Gaëlle Calvary (LIG), Joëlle Coutaz (LIG), Emeric Fontaine (LIG), Teresa Colombi (LudoTIC), Aurore Russo (LudoTIC), Léonid Synyukov (LudoTIC), Stéphane Lavirotte (I3S), Gaëtan Rey (I3S), Jean Yves Tigli (I3S)*
Coordinateurs :	Joëlle Coutaz

Sommaire

1 Introduction

1.1 Rappel des objectifs de la tâche 4 : mettre l'utilisateur dans la boucle

La tâche 4, « Mettre l'utilisateur dans la boucle », a pour objectif de définir les techniques d'interaction et les mécanismes logiciels qui permettent de concilier l'autonomie du système et la nécessité, pour l'utilisateur final, de contrôler ses espaces ambiants. Cette tâche comprend trois sous-tâches :

- Définition de nouvelles techniques d'interaction permettant à l'utilisateur de provoquer une adaptation ou de programmer *a priori* les comportements adaptatifs du système pour les situations connues (par exemple, les moments clefs de la journée : le lever et le départ au travail, le retour le soir, souvent organisés en tâches routinières [Coutaz 10]). À cet objectif correspond le *livrable D4.1 (T0+15) : Document décrivant les techniques d'interaction pour inspecter l'état du système, en contrôler et comprendre l'évolution.*
- Identification des points de contrôle nécessaires à l'utilisateur pour intervenir dans le processus d'adaptation. Cette sous-tâche doit donner lieu au *livrable D4.2 (T0+21) : Document spécifiant les points de contrôle et les IHM correspondantes*.
- Prototypage de ces techniques d'interaction avec le *livrable D4.3 : Document et logiciel du démonstrateur (T0+12, T0+24)*.

1.2 Structure du document

Suite aux travaux menés en collaboration étroite sur les aspects architecturaux de la tâche 2, le consortium a convenu de regrouper les livrables D4.1 et D4.2 et d'utiliser une approche de prototypage rapide tout au long du projet pour les techniques d'interaction (D4.3).

Ce document présente le concept fédérateur de *méta-IHM* que le LIG propose pour décrire le problème du contrôle utilisateur dans les systèmes ambiants et pour analyser les solutions actuelles de l'état de l'art. Une approche générale, *le end-user programming/end-user dévelopment*, est ensuite étudiée en détail avant de présenter nos premières propositions de méta-IHM pour CONTINUUM : l'identification des points de contrôle dans l'architecture suivie des méta-IHM correspondantes illustrées avec le scénario prospectif.

2 Le concept de méta-IHM

2.1 Motivation

Si l'informatique ambiante offre de superbes opportunités, elle est susceptible également d'apporter son lot de nouvelles difficultés. Et l'utilisateur de s'interroger ainsi : « *Comment m'adresser au système ? Le système sait-il que je m'adresse à lui ? Est-il prêt à interpréter mes actions ? Exécute-t-il les actions attendues ? Comment éviter les erreurs ? Etc.* [Bellotti 02] »

En vérité, ces questions valent aussi pour l'informatique conventionnelle que Norman conceptualise en « distance d'exécution » et « distance d'évaluation ». Ces distances désignent les efforts cognitifs et articulatoires (ou physiques) que l'utilisateur doit faire pour exprimer ses objectifs et vérifier qu'ils sont atteints [Norman 86]. Pour Shneiderman, l'un des promoteurs de la manipulation directe, l'utilisateur doit pouvoir garder le contrôle [Shneiderman 87]. Le défi que pose l'informatique ambiante est de donner corps à ces principes dans un milieu extrêmement flexible dont les frontières, en raison de la miniaturisation et de la généralisation du sans fil, sont de moins en moins palpables.

En informatique conventionnelle, le *Shell* d'Unix ou le *Desktop* (bureau) de Windows ou de l'iPhone, permettent de configurer et de contrôler un espace interactif bien cerné, confiné au sein d'une machine et constitué du système d'exploitation et d'un ensemble limité d'applications. En informatique ambiante, l'équivalent du Desktop reste une question ouverte. L'absence de réflexion générale sur le problème de la configuration et du contrôle des espaces interactifs ambiants par l'utilisateur final justifie cette étude. Nous répondons à la diversité des propositions de l'état de l'art par un terme fédérateur : celui de méta-IHM.

2.2 Définition

Une *méta-IHM* recouvre l'ensemble des fonctions (et leur Interface Homme-Machine, IHM) nécessaire et suffisant pour permettre à l'utilisateur de configurer, contrôler et évaluer l'état de son espace interactif ambiant. Un espace interactif ambiant est constitué d'un ensemble de services métiers qui s'exécutent sur un ensemble dynamique et reconfigurable de ressources de calcul, de communication et d'interaction. Parce qu'une méta-IHM est au-dessus des services métiers de l'espace, elle revêt un caractère *méta*. Parce que la méta-IHM permet de configurer, de contrôler et d'évaluer l'état de cet espace, elle a une mission fondamentalement *IHM* [Coutaz 06].

À titre d'illustration, voici deux exemples représentatifs de l'état de l'art : JigSaw [Rodden 04] et tranSticks [Rekimoto 05]. Jigsaw permet à l'utilisateur de programmer son espace par assemblage de pièces de puzzle où chaque pièce représente un objet de l'espace. Par exemple, la figure 1a) montre un assemblage de trois pièces désignant respectivement la sonnette, l'appareil photo et le PDA. Ce programme signifie que : *« lorsque quelqu'un sonne à la porte, le prendre en photo et afficher la photo sur le PDA ».* Comme le montre la figure 1b), un programme Jigsaw est construit sur une tablette PC au moyen d'un éditeur dédié.

a) b)

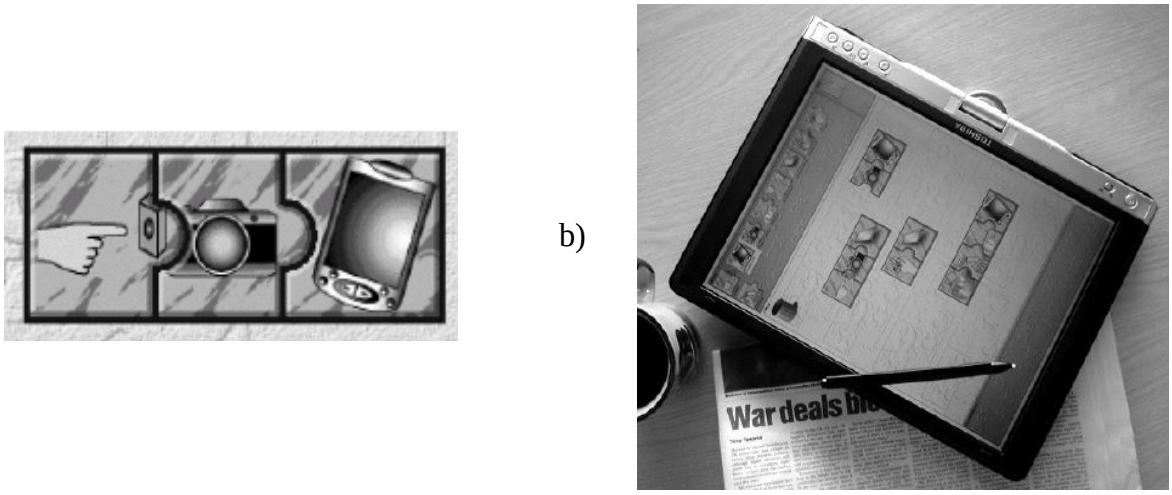


Figure 1. Jigsaw. En a) un programme résultant de l'assemblage de 3 pièces de puzzle. En b) l'éditeur dédié. ([Rodden 04])

L'utilisateur de Jigsaw configure son espace par manipulation d'entités numériques (les pièces de puzzle) qui représentent des objets physiques (la sonnette, l'appareil photo, le PDA). Avec tranSticks, la situation est inverse : l'utilisateur configure son espace en manipulant des entités physiques (les tranSticks) qui représentent des connexions numériques. Un tranStick est une clef USB dotée d'un numéro que partage un autre tranStick. Comme le montre la figure 2, une paire de tranSticks représente une connexion réseau potentielle. L'utilisateur reconnaît une paire par la même étiquette que portent les deux tranSticks. Lorsque les tranSticks sont branchés sur deux dispositifs (PC, Vidéoprojecteurs, et de manière générale, tout dispositif doté d'une adresse IP), les connexions effectives sont établies entre les deux entités.

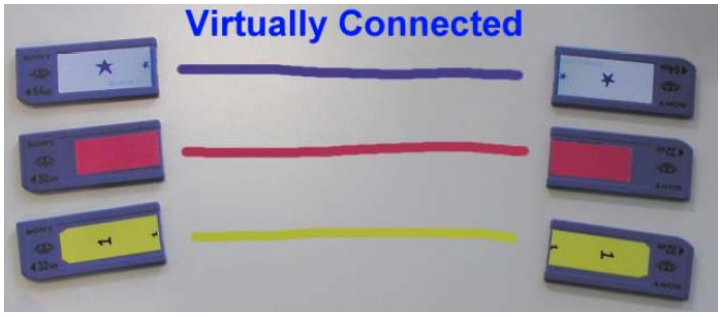


Figure 2. Trois paires de tranSticks. ([Rekimoto 05])

Ces deux exemples montrent deux approches opposées au problème de la configuration d'espaces interactifs ambiants. L'état de l'art en révèle beaucoup d'autres, mais développés au cas par cas et, en général, sans lien avec une infrastructure d'exécution comme celle de CONTINUUM. Cette situation nous a conduits à définir les dimensions caractérisantes d'une méta-IHM qui, ensemble, forment un espace taxonomique. Cet espace nous sera ensuite utile pour analyser l'état de l'art, en relever les insuffisances et justifier de nouvelles recherches.

2.3 Espace taxonomique

Comme tout service numérique, une méta-IHM :

- permet de manipuler des *entités*,
- offre des fonctions dont la *puissance* est censée couvrir l'utilité attendue,
- a des *qualités* en réponse à des requis d'utilisabilité.

La figure 3 déploie ces trois aspects avec les cadrans *entités*, *puissance* et *qualités*, que nous affinons en six axes non orientés :

- Pour le cadran *entités manipulées* : nature des entités et type de manipulation de ces entités,
- Pour le cadran *puissance* : services offerts et extensibilité du langage d'interaction,
- Pour le cadran *qualités* : niveau d'intégration et niveau de contrôle laissé à l'utilisateur.

Dans les sections qui suivent, nous présentons chacun de ces axes en détail illustrés par des exemples de l'état de l'art ou par des scénarios ou prototypes de CONTINUUM. *Cet espace de classification est une première proposition susceptible d'être révisée au cours du projet*.

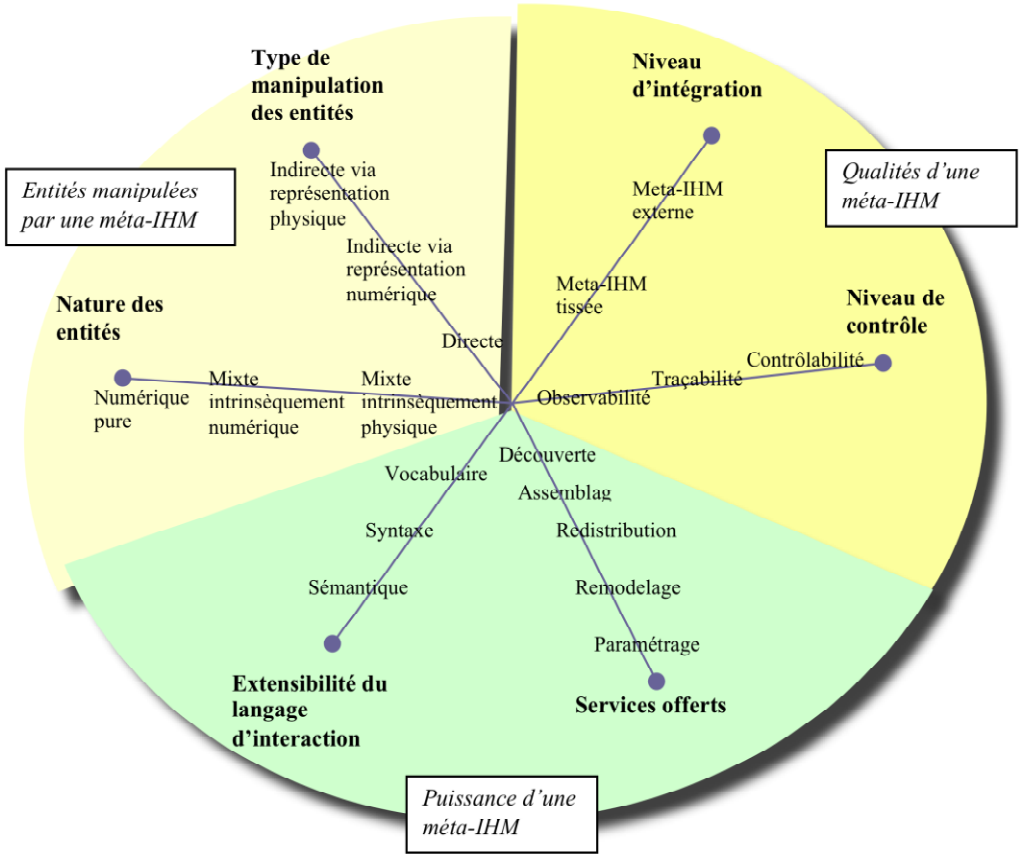


Figure 3. Espace taxonomique pour l'étude de méta-IHM.

2.4 Entités manipulées par une méta-IHM

Dans les Desktops conventionnels, les concepts manipulés sont des entités numériques graphiques (icônes, fenêtres, etc.) dont la manipulation s'effectue au moyen d'un instrument de pointage ou du clavier. Dans un milieu ambiant, au-delà des entités numériques, toute entité physique peut être manipulée et ceci de multiples façons. Nous convenons donc de deux axes pour caractériser les entités manipulées par une méta-IHM : la nature des entités manipulables et comment ces entités sont manipulées (c.-à-d. le type de manipulation).

2.4.1 Nature des entités manipulables

Les entités manipulables par une méta-IHM sont de trois types : les entités numériques pures et les entités mixtes.

- Comme pour un Desktop conventionnel, une méta-IHM permet d'agir sur l'état d'*entités numériques pures* qui servent à présenter les services métiers et système : fenêtres et icônes notamment.

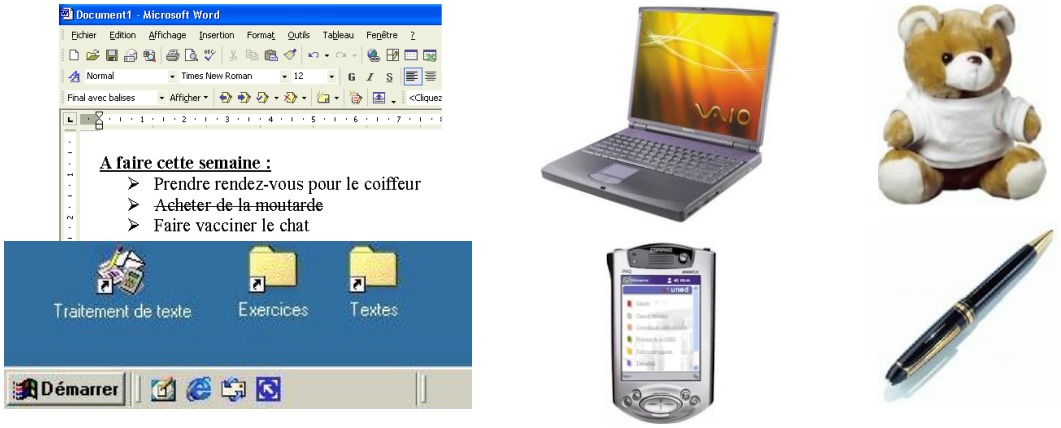


Figure 4. a) Entités numériques pures (icônes, fenêtres) b) entités mixtes intrinsèquement numériques (PC, PDA) c) Entités mixtes intrinsèquement physiques (stylo communicant, peluche augmentée d'un service de journal intime).

- Une *entité mixte* est le résultat d'un couplage (permanent ou non) entre une entité physique et une entité numérique (concept ou fonction) [Coutrix 05]. De nombreux travaux visent à caractériser les entités mixtes : la taxonomie de Fitzmaurice est limitée au seul cas des briques [Fitzmaurice 95], celle de Fishkin est fondée sur les métaphores de nom et de verbe [Fishkin 04], et Holmquist caractérise les objets mixtes— ou *tokens*, en fonction de leur rôle (conteneur, représentant, outil) [Holmquist 99]. Toutefois, ces taxonomies ne reflètent pas la distinction entre entités mixtes intrinsèquement numériques de celles qui sont intrinsèquement physiques. Aussi, nous proposons le distinguo suivant :
    - *Une entité mixte est intrinsèquement numérique* si, pour être utile, elle nécessite une composante numérique : les dispositifs PC, PDA, smartphones et leurs périphériques relèvent de cette catégorie. Sans leur dimension numérique, ils perdent leur raison d'être.
    - *Une entité mixte est intrinsèquement physique* si elle ne perd pas sa raison d'être lorsqu'on lui retire son amplification numérique : un stylo augmenté, mais sans son tag RFID ou sa connectivité au réseau, peut toujours servir à écrire. Il conserve sa raison d'être. Il en va de même pour un réfrigérateur augmenté (exemple célèbre du réfrigérateur Electrolux).

La figure 4 illustre en images les trois catégories d'entités manipulables par une méta-IHM.

La distinction que nous introduisons entre entités mixtes intrinsèquement numériques et entités mixtes intrinsèquement physiques tient à l'absence de frontières nettes en informatique ambiante. Dès lors, comment l'utilisateur peut-il distinguer les objets physiques reliés à son espace de ceux qui en sont exclus? Par définition, les entités mixtes intrinsèquement numériques peuvent être comprises comme éléments de l'espace interactif puisqu'elles ont été prévues pour : sans support numérique, elles ne servent à rien. Inversement, l'*affordance perçue*[1] des entités intrinsèquement physiques ne suggère pas nécessairement leur appartenance à l'espace. Si l'utilisateur n'y prend garde, le stylo communicant ou la peluche augmentée

[1] Le concept d'*affordance* a été introduit par Gibson, puis repris par Norman [Norman 99], pour désigner la capacité d'un objet à forcer chez l'utilisateur son bon usage à partir de la seule perception de cet objet. Par exemple, une poignée de porte, force, par sa seule forme perçue, l'action de saisir avec la main.

risquent d'être vus comme des objets du quotidien déconnectés de l'espace. Dès lors, l'utilisateur non conscient de la connectivité de ces objets risque de dévoiler, sans intention, des informations personnelles. Les entités mixtes intrinsèquement physiques (comme la clef imaginée dans le cadre du scénario industriel de CONTINUUM) nécessitent donc une attention toute particulière de la part des concepteurs.

2.4.2 Type de manipulation des entités

Depuis l'apparition des interfaces tangibles, on distingue trois types de manipulation d'entités :

- Manipulation de l'entité proprement dite (non pas d'un représentant). Il s'agit de manipulation vraiment directe.
- Manipulation d'un représentant physique de l'entité.
- Manipulation d'un représentant numérique de l'entité.

Une méta-IHM peut inclure une ou plusieurs de ces techniques, les caractères direct et indirect ayant un impact sur la nature de l'interaction [Beaudouin-Lafon 97]. La figure 5 illustre en images les trois types de manipulation.

a) b) c)

Figure 5. a) Dynawall ([Streitz 99]): manipulation directe d'entités b) Jigsaw ([Rodden 04]): manipulation d'entités par le biais d'un représentant physique c) Jigsaw : Manipulation d'entité par le biais d'un représentant numérique.

Par exemple, dans le Dynawall [Streitz 99] l'utilisateur peut déplacer les fenêtres sur une grande surface d'affichage formée de trois écrans placés côte à côte. La migration de fenêtres (entité numérique) se fait sur la fenêtre elle-même au moyen du doigt. C'est de la manipulation directe de fenêtres. Dans Jigsaw [Rodden 04], l'utilisateur programme son environnement par assemblage de pièces de puzzle, chacune représentant une entité de l'environnement : l'utilisateur manipule les entités de l'environnement par indirection (quand bien même il manipule directement les représentants). L'utilisateur (ou le concepteur) a le choix entre deux possibilités : soit les pièces de puzzle sont des entités physiques. Alors, l'utilisateur manipule un représentant physique des entités de l'environnement ; soit les pièces sont des entités numériques et l'utilisateur manipule un représentant numérique des entités de l'environnement.

2.5 Puissance d'une méta-IHM

La puissance d'une méta-IHM réside dans l'étendue des services qu'elle offre, mais aussi dans l'extensibilité du langage d'interaction mis à la disposition des utilisateurs.

2.5.1 Services génériques d'une méta-IHM

Les concepteurs du Desktop du Xerox Star [Canfield Smith 82] ont su mettre à profit la notion de commande générique (copier, couper, coller, déplacer, etc.) ancrée aujourd'hui dans tout système interactif. En

application de ce principe fondateur, une méta-IHM devrait, en raison de son rôle de « Desktop du futur », offrir également des services génériques.

En l'état de nos investigations, nous proposons les services suivants : *assemblage*, *redistribution* et *remodelage* que nous tenons de notre expérience en plasticité des IHM [Calvary 01], mais aussi *découverte* et *paramétrage* que nous tenons des recommandations du projet Speakeasy [Newman 02] et de son successeur Obje [Edwards 09]. En fonction des capacités d'autonomie de la méta-IHM, ces services sont soumis ou non au *contrôle de l'utilisateur*, propriété que nous développons plus loin en 2.6.

Assemblage (désassemblage) : L'assemblage permet de construire un espace ambiant par couplage d'objets [Barralon 05]. C'est le cas de Jigsaw et de tranSticks, mais aussi des tablettes d'Hinckley [Hinckley 00b] où l'utilisateur doit choquer deux écrans pour les assembler (voir la figure 6). A contrario, dans Aris [Biehl 04], les objets, des ressources de calcul, d'interaction et d'affichage, sont assemblés d'office. L'utilisateur n'a pas le contrôle de l'assemblage.

Figure 6. Les tablettes d'Hinckley. Le choc entre deux tablettes entraîne leur assemblage. ([Hinckley 00b])

Distribution (redistribution) : La distribution consiste à allouer des entités numériques à des objets mixtes. Redistribuer, c'est modifier cette allocation. Par exemple, dans les scénarios *suis-moi*, l'IHM numérique de services métiers migre vers un objet mixte compatible à proximité de l'utilisateur. Par exemple, dans Aura [Sousa 03], lorsque l'utilisateur se trouve près de la machine à café, un nouveau mail est alloué automatiquement sur l'afficheur de la machine. Ou encore, une IHM peut être répartie entre un PDA qui fait office de télécommande et un mur d'images qui sert de surface partagée. Comme le montre Pick and Drop de la figure 7, une zone de dessin est allouée sur un grand écran tandis que le PDA contient la palette d'outils [Rekimoto 98].

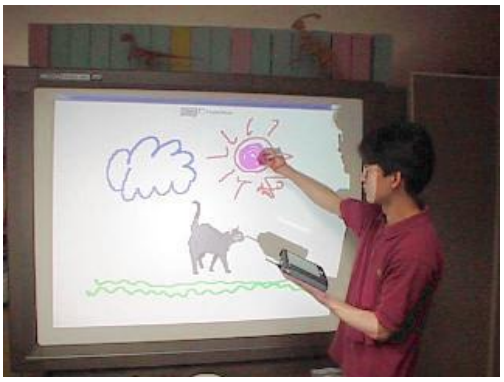


Figure 7. Le Pick and Drop de Rekimoto. L'utilisateur dessine sur un écran public alors que la palette d'outils pour le dessin est disposée sur un PDA. ([Rekimoto 98])

Remodelage : Le remodelage consiste à modifier l'IHM d'un service métier par substitution, ajout ou suppression d'interacteurs, ou par leur simple réorganisation [Calvary 01]. Typiquement, la migration (redistribution) de l'IHM graphique d'un PC vers un PDA nécessite un remodelage en raison du changement de taille des écrans. Collapse-to-Zoom, qui permet à un utilisateur d'afficher ou de supprimer des parties de pages web, est un exemple de remodelage sous le contrôle de l'utilisateur [Baudisch 04]. Plastic Clock de la figure 8 se remodèle en fonction de la surface d'affichage disponible. En l'absence d'écran, des interacteurs vocaux se substituent aux interacteurs graphiques.

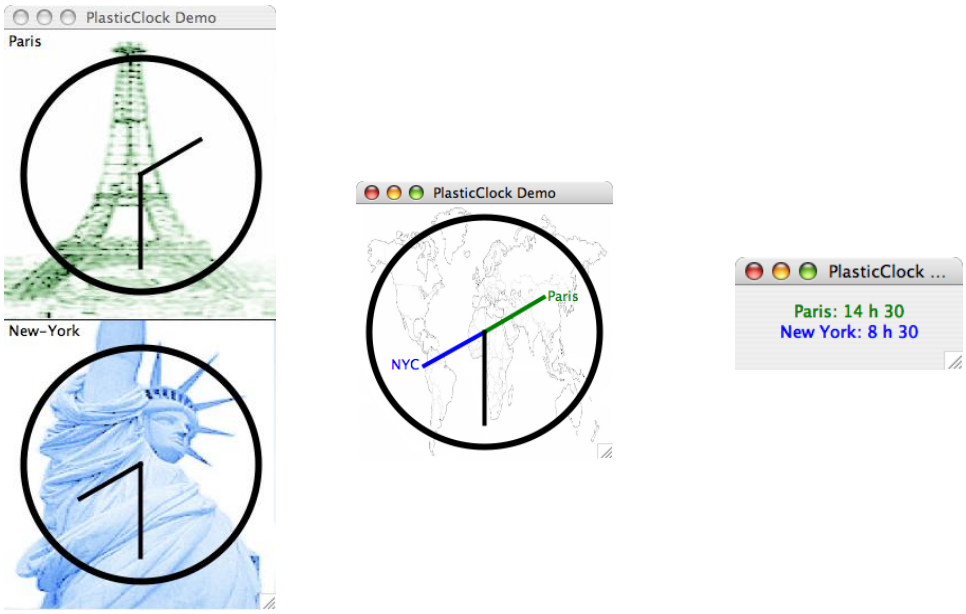


Figure 8. Plastic Clock. À gauche, la surface d'affichage est suffisante pour afficher l'heure de New York et de Paris de manière analogique (aiguilles de montre et images représentatives des deux villes). Au centre, un interacteur fusionne les heures des deux villes avec l'ajout d'une aiguille. À droite, la version minimaliste.

Découverte : Puisque les frontières d'un espace ambiant sont flexibles et impalpables, une méta-IHM doit fournir un service de découverte. La découverte avec filtrage permet d'affiner les besoins. Dans Speakeasy [Newman 02] un browser (figure 9) liste l'ensemble des objets disponibles (fichiers, services métiers et projecteurs vidéo) répondant à des critères de lieu et d'appartenance : dans la salle, dans le bâtiment, ou appartenant à une personne précise.

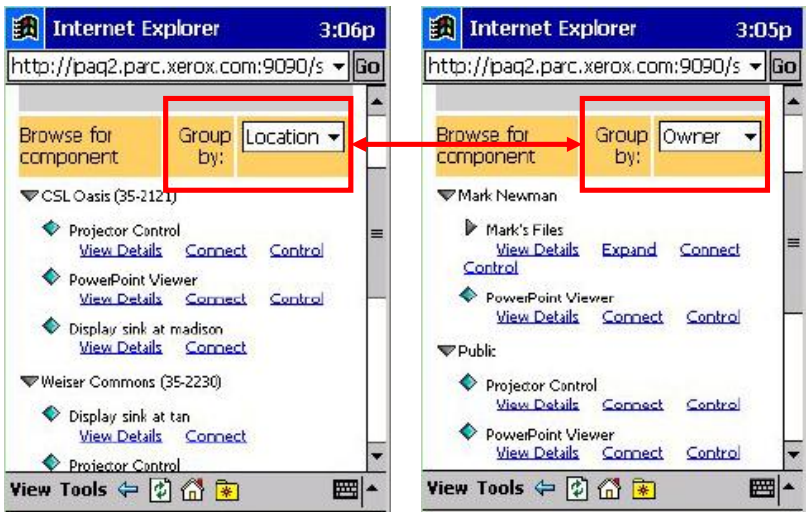


Figure 9. Speakeasy. En encadré, les critères de découverte. À gauche, le filtre de localisation, à droite, l'appartenance à une personne. ([Newman 02])

Paramétrage : Le paramétrage désigne la capacité, pour l'utilisateur, de contrôler un objet de l'espace, à la manière des préférences et valeurs par défaut du Desktop actuel. Par exemple, un projecteur peut configurer sa résolution d'affichage en fonction de l'ordinateur auquel il est connecté. Ce paramétrage peut être surchargé par l'utilisateur qui peut ainsi programmer son environnement au niveau de finesse désiré. La figure 10 montre un exemple de paramétrage tiré du projet E-Gadget [Marcopoulos 04] : l'ouverture du livre provoque l'allumage de la lampe du salon.

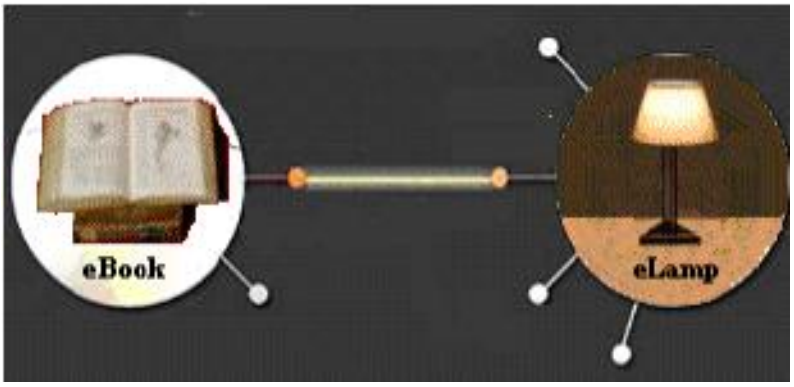


Figure 10. E-Gadget. Un *eBook* est relié à une *e-lamp* en sorte que la lampe s'allume dès l'ouverture du livre. ([Marcopoulos 04])

2.5.2 Extensibilité du langage d'interaction

Une méta-IHM regroupe un ensemble de services (et notamment les services génériques de base que nous venons de décrire). Pour que ces services soient accessibles à l'utilisateur, ils doivent être dotés d'une IHM. Toute IHM se décrit par un langage d'interaction (ou ensemble de techniques d'interaction).

Comme tout langage, le langage d'interaction d'une méta-IHM est défini par l'ensemble des phrases conformes à une grammaire donnée : vocabulaire (notamment les objets manipulés), syntaxe (règles d'agencement des mots du vocabulaire) et sémantique (le, ou les services demandés parmi ceux proposés par la méta-IHM).

L'extensibilité du langage d'interaction de la méta-IHM va de pair avec le caractère dynamique des espaces ambiants. Or, une méta-IHM peut être limitée en vocabulaire, notamment par la couverture des entités manipulées. Par exemple, le vocabulaire d'Aris [Biehl 04] est réduit aux classes d'objets attendus dans une salle : murs, mobiliers, dispositifs d'affichage et services numériques. Par conséquent, cette méta-IHM ne peut être exportée dans un environnement différent de celui prévu. Inversement, CAMP permet l'extensibilité du vocabulaire [Truong 04]. Une méta-IHM peut être limitée en syntaxe : le système ne sait pas apprendre de nouvelles constructions sémantiquement équivalentes. Elle peut être limitée en sémantique si l'ensemble des services est figé par conception. Par exemple, CubeTV [Block 04], en figure 11, n'est destiné qu'à la sélection de chaînes de télévision (donc limitée en sémantique), grâce à un cube tangible que l'on oriente en fonction des chaînes souhaitées. La syntaxe est ici limitée par les manipulations possibles sur le cube.

Figure 11. CubeTV. L'utilisateur change les chaines de la télévision grâce au cube qu'il tient dans ses mains. ([Block 04])

2.6 Qualité d'une méta-IHM

Au-delà des qualités ergonomiques et logicielles usuelles, nous avons regroupé sous ce terme le niveau de contrôle qu'une méta-IHM accorde à l'utilisateur et son niveau d'intégration parmi les services métiers et système qu'elle permet de configurer et superviser.

2.6.1 Niveaux de contrôle offerts par une méta-IHM

En Interaction Homme-Machine, nous disposons de nombreux guides ergonomiques. Nous retenons les propriétés d'observabilité [Gram 96], de traçabilité et de contrôlabilité pour leur capacité à couvrir de manière synthétique les principes de Norman et de Shneiderman évoqués en introduction. L'observabilité est la capacité du système à rendre perceptible l'état pertinent du système (c'est un feedback approprié, ni en trop, ni en moins). Avec l'observabilité, l'utilisateur peut évaluer l'état du système. Avec la traçabilité, l'utilisateur peut évaluer l'évolution du système : c'est l'observabilité d'une suite d'états pertinents. Typiquement, la barre de progression est un interacteur qui sert la traçabilité. La contrôlabilité est la capacité donnée à l'utilisateur de modifier le cours des choses. Il n'est plus simple observateur, mais acteur.

Le tableau 1 met en regard les *niveaux de contrôle* possibles sur les *services* d'une méta-IHM.

	Découverte	**Assemblage**	**Redistribution**	**Remodelage**	**Paramétrage**
Observabilité	L'utilisateur peut percevoir les objets manipulables et leur état	L'utilisateur peut percevoir les assemblages et leur état	L'utilisateur peut percevoir les redistributions et leur état	L'utilisateur peut percevoir les remodelages et leur état	L'utilisateur peut percevoir les paramétrages et leur état
Traçabilité	L'utilisateur peut suivre l'évolution de l'état de la découverte mais ne peut pas intervenir	L'utilisateur peut suivre l'évolution de l'état d'assemblages mais ne peut pas intervenir	L'utilisateur peut suivre l'évolution de l'état d'une redistribution mais ne peut pas intervenir	L'utilisateur peut suivre l'évolution de l'état d'un remodelage mais ne peut pas intervenir	L'utilisateur peut suivre l'évolution de l'état de paramétrages mais ne peut pas intervenir
Contrôlabilité	L'utilisateur peut filtrer la découverte	L'utilisateur peut prendre en main l'assemblage	L'utilisateur peut prendre en main la redistribution	L'utilisateur peut prendre en main le remodelage	L'utilisateur peut modifier le paramétrage

Tableau 1. Services d'une méta-IHM et les niveaux de contrôle offerts.

2.6.2 Niveaux d'intégration d'une méta-IHM

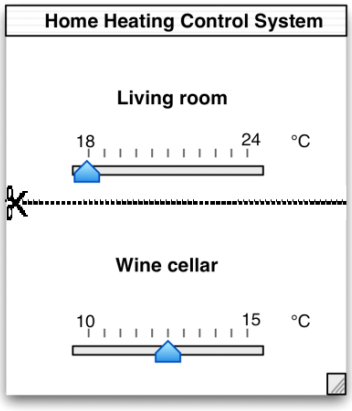
Nous voyons trois lignes directrices à l'intégration des méta-IHM dans les espaces ambiants : soit l'IHM de la méta-IHM est externe aux IHM des services métiers et système, soit elle est tissée avec elles, soit elle est en partie externe et tissée.

Tissée : Une méta-IHM est tissée lorsque son IHM est intégrée dans l'IHM des services métiers et système. Prenons l'exemple de la figure 12a) qui représente l'IHM graphique de réglage de la température du domicile. Dans cette fenêtre, on constate la présence d'une paire de ciseaux et de pointillés qui indique que la fenêtre est sécable et donc redistribuable. Dans cet exemple, le service redistribuer de la méta-IHM est représenté par des entités numériques analogiques (paire de ciseaux et pointillés) intégrés dans l'IHM du service métier.

Externe : Une méta-IHM est externe lorsque son IHM est totalement indépendante des IHM des services métiers. Sur la figure 12b) est présenté l'exemple de Aris [Biehl 04] : dans une fenêtre dédiée, sont présentés sous forme graphique, l'environnement (murs, mobiliers, dispositifs d'affichage) ainsi que les miniatures des services métiers disponibles dans l'espace ambiant. Pour redistribuer l'IHM d'un service métier, l'utilisateur déplace la miniature du service en question sur la représentation d'un objet mixte jouant le rôle de surface d'affichage.

Mixte : Une méta-IHM peut être à la fois tissée et externe lorsqu'elle combine les propriétés caractéristiques des IHM tissée et externe.

a)



b)

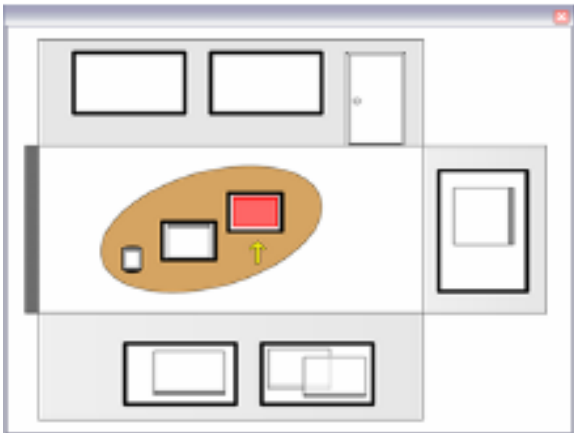


Figure 12. a) Une méta-IHM tissée, b) La méta-IHM externe d'Aris ([Biehl 04])

Ayant présenté les éléments de notre classification, nous sommes maintenant en mesure d'analyser de manière systématique les exemples de l'état de l'art les plus représentatifs.

3 Analyse de l'état de l'art en matière de méta-IHM

Les tableaux 2 et 3 synthétisent les résultats de notre étude de l'état de l'art. Nous les analysons en détail selon les axes de notre taxonomie, puis nous en tirons les leçons pour notre propre conception.

	ENTITES MANIPULEES							
	Nature			Manipulation			Niveau d'intégration	
	Mixte phys.	Mixte num.	Num.	Directe	Indirecte repr. num.	Indirecte repr. phys.	Méta-IHM tissée	Méta-IHM externe
ARIS [Biehl 04]		X	X		X			X
aCAPella [Dey04]	X	X	X		X			X
DongleRelate [Kortuem05]		X			X			X
MigreXML [Molina06]		X	X		X			X
AmbientDesktop [Barralon05]		X	X	X	X			X
SpeakEasy [Newman02]		X	X		X			X
JigSaw [Rodden04]	X	X	X		X	X		X
e-Gadget [Marcopoulos04]	X	X			X			X
ICAP [Sohn06]	X	X	X		X			X
LegoLogo [McNerney04]	X	X				X		X
Bope [Pering05]		X	X		X	X		X
MightyMouse [Booth02]		X	X		X			X
Dynamo [Izadi03]		X	X		X			X
Peebles [Myers02]		X	X		X			X
PutThatThere [Harada03]			X	X	X			X
IStuff [Ballagas 03]	X	X		X				X
ICrafter [Ponnekanti01]		X	X		X			X
Collapse-to-Zoom [Baudisch04]			X	X			X	
AttachMe [Grolaux05]			X	X			X	
Stitching [Hinckley00]			X	X	X			X

CubeTV [Block04]			X			X		X
DataTiles [Rekimoto03]			X		X	X		X
Triangles [Gorbet98]		X	X		X	X		X
Mediablock [Ullmer98]			X			X		X
SyncTap [Rekimoto03]		X	X	X	X			X
Transticks [Ayatsuka05]		X	X			X		X
Oscar [Newman08]		X	X		X			X
CAMP [Truong04]	X	X	X		X			X
Media Cubes [Blackwell04]	X	X	X	X		X		X

Tableau 2. Synthèse de l'état de l'art : dimensions « Entités manipulées » et « Niveau d'intégration » de l'IHM de la méta-IHM dans l'IHM des applications.

	SERVICES ET NIVEAU DE CONTROLE														
	Découverte			Assemblage			Redistribution			Remodelage			Paramétrage		
	O	T	C	O	T	C	O	T	C	O	T	C	O	T	C
ARIS [Biehl 04]	X						X	X	X				X		
aCAPella [Dey04]	X												X		X
DongleRelate [Kortuem05]	X		X	X		X									
MigreXML [Molina06]	X						X	X	X						
AmbientDesktop [Barralon05]	X			X		X	X		X						
SpeakEasy [Newman02]	X		X						X						
JigSaw [Rodden04]	X			X		X							X		X
e-Gadget [Marcopoulos04]	X			X		X							X		X
ICAP [Sohn06]	X		X												X
	Découverte			Assemblage			Redistribution			Remodelage			Paramétrage		
	O	T	C	O	T	C	O	T	C	O	T	C	O	T	C
LegoLogo [McNerney04]				X		X							X		
Bope [Pering05]						X			X						
MightyMouse [Booth02]	X								X						

Dynamo [Izadi03]	X												X		X
Peebles [Myers02]X							X		X						
PutThatThere [Harada03]									X						
IStuff [Ballagas 03]															X
ICrafter [Ponnekanti01]	X		X												X
Collapse-to-Zoom [Baudisch04]												X			
AttachMe [Grolaux05]						X			X						X
Stitching [Hinckley00]						X	X	X	X						
CubeTV [Block04]															
DataTiles [Rekimoto03]				X		X			X						
Triangles [Gorbet98]				X		X									X
Mediablock [Ullmer98]															X
SyncTap [Rekimoto03]						X									
Transticks [Ayatsuka05]						X									
Oscar [Newman08]	X		X			X			X						
CAMP [Truong04]	X			X		X							X		X
Media Cubes [Blackwell04]	X			X		X							X		X

Tableau 3. Synthèse de l'état de l'art : dimensions services offerts et niveau de contrôle de ces services.

3.1 Entités manipulées et leurs manipulations

Les colonnes *Nature* du sous-tableau *Entités manipulées* du tableau 2 révèlent que les méta-IHM de l'état de l'art permettent de manipuler plusieurs types d'entités. On assiste en majorité à des méta-IHM manipulant à la fois des entités numériques pures et des entités mixtes intrinsèquement numériques, suivies par des méta-IHM ne traitant que des entités numériques. Rares sont les méta-IHM consacrées aux entités mixtes (qu'elles soient intrinsèquement numériques ou intrinsèquement physiques). Pour ces méta-IHM, la manipulation des entités mixtes se fait en majorité par l'intermédiaire de représentations numériques alors que, au nom des principes des TUI (Tangible User Interfaces), on aurait pu s'attendre à l'exploitation massive de la manipulation directe de la composante matérielle de ces entités.

Pour les méta-IHM qui ne manipulent que des entités numériques pures, c'est la manipulation directe qui prime, en conformité avec les principes de la manipulation directe du style WIMP (Window Icon Menu Pointing).

En synthèse, la manipulation directe s'impose dans les méta-IHM à entités numériques pures et la manipulation indirecte est le fait des méta-IHM qui incluent des entités mixtes. De plus, quand il y a indirection, le numérique l'emporte sur le physique. Une explication possible est l'encombrement de l'environnement que provoqueraient des objets matériels supplémentaires. Pour CONTINUUM, nous reprenons à notre compte cette hypothèse.

3.2 Services et qualités ergonomiques

Considérons maintenant le sous-tableau *Services et Niveaux de contrôle* du tableau 2. Aucune méta-IHM ne couvre l'ensemble des services génériques de notre taxonomie. L'assemblage de ressources est le service le plus fréquent suivi par le trio redistribution, paramétrage et découverte. Le remodelage est quasi absent.

Dans les exemples étudiés, l'assemblage et la redistribution vont souvent de pair. Mais l'assemblage est également proposé avec le paramétrage comme dans E-Gadget [Marcopoulos 04] où l'utilisateur peut assembler et paramétrer une lampe et un livre pour que la lumière s'allume quand le livre est ouvert. La découverte est souvent couplée à l'assemblage ou à la redistribution pour permettre à l'utilisateur d'organiser les entités de l'espace ambiant. Le remodelage, largement derrière les autres services, est un problème difficile qui relève de la plasticité des IHM pour lequel il n'existe pas encore de solution générale, ce qui explique sa rareté.

La qualité d'observabilité est très bien respectée, ce qui est cohérent avec la raison d'être d'une méta-IHM : montrer à l'utilisateur l'ensemble des entités en présence. Les services d'assemblage, de redistribution et de paramétrage respectent ce critère. En l'absence d'observabilité, l'utilisateur risque de rencontrer des difficultés au moment de la prise en mains. Par exemple, « Attach Me, Detach Me, Assemble Me like You Work » [Grolaux 05] permet l'assemblage, la redistribution et le remodelage mais ne prend pas en compte le besoin de l'utilisateur de comprendre la marche à suivre pour les utiliser.

Le critère de traçabilité n'est pris en compte que pour le service de redistribution. L'état de l'art ne révèle aucun exemple gérant la traçabilité pour les autres services. Ce détail mérite alors un approfondissement futur. Les méta-IHM respectant cette qualité pour le service de redistribution mettent en œuvre des animations qui permettent à l'utilisateur de suivre l'état de la redistribution. Notamment dans Aris [Biehl 04], l'utilisateur peut redistribuer une fenêtre entre deux surfaces d'affichage grâce aux miniatures de ces fenêtres, déplaçables dans un éditeur. Parallèlement à cette manipulation, un fantôme de la fenêtre fait aussi le chemin entre les deux surfaces d'affichage, rendant traçable la redistribution dans le monde physique. L'animation semble un bon moyen de rendre traçables les actions de l'utilisateur, en particulier pour la redistribution et le remodelage.

La contrôlabilité est généralement respectée pour les services d'assemblage, de redistribution et de paramétrage, en conformité avec le rôle attribué à une méta-IHM. A contrario, les services de découverte et de remodelage sont peu contrôlés. Le service de découverte avec contrôle permet de filtrer par lieu ou appartenance les entités de l'environnement. La plupart des méta-IHM détourne le problème en contraignant l'utilisation de la méta-IHM à un lieu clos, le service de découverte par filtrage étant donc souvent omis. Le remodelage est rarement contrôlé contrairement aux principes fondateurs de l'Interaction Homme-Machine.

3.3 Intégration des méta-IHM dans les IHM métiers et système

Le tableau 2 indique qu'en majorité les méta-IHM sont de type externe. Deux exceptions méritent d'être relevées : « Collapse to zoom » [Baudisch 04] et « Attach Me, Detach Me, Assemble Me like You Work » [Grolaux 05]. Toutes deux permettent à l'utilisateur d'encadrer au stylet ou à la souris, des morceaux d'IHM de service métier pour les supprimer, les déplacer, ou les modifier. Toutefois, rien dans l'IHM ne permet à l'utilisateur de deviner l'existence de ces services (le remodelage n'est pas observable a priori). Les IHM tissées de notre état de l'art sont rarement observables, sans doute pour ne pas surcharger les IHM des services métiers ou pour ne pas risquer d'être incohérentes avec le style de ces IHM.

Le tableau 2 révèle une forte dépendance entre les propriétés d'externalité et d'observabilité. En effet, une méta-IHM externe non observable comme celle de Stitching [Hinckley 00a], a peu de chance d'être découverte. Dans Stitching, inspiré des gestes synchronisés, l'utilisateur fait appel à deux services de méta-IHM en un seul geste continu au stylet : assembler deux tablettes et redistribuer des entités numériques entre ces tablettes. L'interaction est efficace en termes de trajectoire, les effets du geste sont observables pendant l'action et *a posteriori*, mais si le geste est inconnu, rien dans l'espace ambiant n'indique cette possibilité. C'est aussi le problème des *smart rooms* où des agents intelligents, fondés sur les principes de l'interaction implicite[2], assurent des services de méta-IHM sur la seule reconnaissance des activités humaines. On assiste ici au difficile problème de l'équilibre entre laisser le contrôle à l'utilisateur au risque de le surcharger de nouvelles tâches et l'autonomie des systèmes au risque de contrarier les intentions de l'utilisateur.

3.4 Extensibilité du langage d'interaction

Les méta-IHM qui offrent le plus de services sont limitées par leur langage d'interaction. Un compromis doit être fait entre couverture des services, puissance d'expression du langage d'interaction et facilité de compréhension pour l'utilisateur.

Typiquement, Aris [Biehl 04] couvre l'assemblage de ressources d'un milieu donné (une salle) et la redistribution des IHM métiers sur ces ressources. L'IHM de la méta-IHM opère par manipulation directe sur des représentants numériques en conformité avec la topologie du monde réel (celui dans lequel l'utilisateur se trouve). Cette manipulation est simple à comprendre. Mais résolument ancrée dans la métaphore du monde réel, elle perd la puissance d'expression du monde numérique. Speakeasy et son successeur Obje, au contraire, utilisent des représentations numériques abstraites (des hiérarchies), moins faciles à comprendre, mais plus puissantes puisque l'utilisateur peut traiter de redistribution et de filtrage sur un espace autre que la salle dans laquelle il se trouve.

3.5 Synthèse

En synthèse, des patrons de méta-IHM semblent émerger comme suit :

- Les méta-IHM de redistribution sont soit externes (comme pour Aris [Biehl 04], Speakeasy [Newman 02], Obje [Edwards 09], Ambient Desktop [Barralon 05]), soit tissées (comme pour « Collapse-to-zoom » [Baudisch 04] et « Attach Me Detach Me Assemble Me like You Work » [Grolaux 05]). Lorsqu'elles satisfont la propriété d'observabilité, elles s'appuient sur une métaphore visuelle en conformité avec la topologie du lieu d'interaction.

[2] L'interaction implicite s'appuie sur l'interprétation, par le système, d'actions que l'utilisateur ne destine pas au système. Par exemple, dans le projet européen FAME, un *topic spotter*, reconnaissant le sujet de la discussion entre utilisateurs, affiche sur un mur des informations en relation avec le sujet d'intérêt.

- Les méta-IHM tangibles telles que LegoLogo [McNerney 04], Triangles [Gorbet 98], Datatiles [Rekimoto 01], ou cube TV [Block 04] visent toujours la manipulation d'entités numériques via des entités physiques. Elles sont souvent externes et ne satisfont pas toujours le critère d'observabilité.
- Certaines méta-IHM d'assemblage sont centrées sur la technique d'interaction plutôt que sur le service lui-même. Les tablettes d'Hinckley [Hinckley 00b] ou les smart-Its [Holmquist 01] illustrent cette catégorie.
- Les méta-IHM qui permettent de programmer l'environnement comme E-gadget [Marcopoulos 04], Jigsaw [Rodden 04], ICAP [Sohn 06], CAMP [Truong 04] ou encore a CAPpella [Dey 04] sont toujours externes et satisfont les propriétés d'observabilité et de contrôlabilité. Ces méta-IHM, qui relèvent du *end-user programming*, font en ce moment l'objet d'études actives.

Au-delà de ces patrons, notre analyse montre que la tendance actuelle est à la méta-IHM externe portant sur un mélange d'entités numériques et d'entités mixtes intrinsèquement numériques, de même aux méta-IHM de programmation.

- Les méta-IHM externes à représentation numérique offrent une bonne observabilité de l'état de l'espace ambiant et de là, les bases pour assurer la traçabilité et la contrôlabilité. Les recherches se penchent aussi sur l'exploitation des principes des IHM tangibles. Elles s'appliquent à concrétiser des entités numériques pures. Toutefois, la conception de méta-IHM tangibles pour espaces ambiants est peut-être prématurée : en 25 ans, l'utilisateur a pris ses habitudes avec la souris et les objets numériques cliquables. Les IHM tangibles appellent un changement de paradigme et donc un changement des habitudes des utilisateurs.
- Aucune méta-IHM n'est à la fois « externe » et « tissée ». Nous estimons qu'il s'agit là d'une piste intéressante de solution sachant que les méta-IHM externes sont plutôt faciles à utiliser au regard des propriétés d'observabilité et que les méta-IHM tissées sont par définition mieux intégrées dans le tissu de nos activités.
- Aucune méta-IHM de programmation d'environnement n'intègre à la fois la redistribution et le remodelage. On constate une réelle dichotomie entre les méta-IHM de programmation et les méta-IHM de redistribution. Leurs fonctionnalités sont disjointes alors que la finalité est la même : configurer l'espace interactif ambiant.

En synthèse, l'analyse de l'état de l'art au regard de notre taxonomie ouvre les pistes de solutions suivantes : (a) explorer l'apport des IHM tangibles mais, pour le court terme conserver le paradigme actuel, (b) mixer tissage (pour l'intégration) et externalité (pour l'observabilité), (c) explorer l'apport du *end-user programming* et du *end-user development* pour sa puissance d'expression. C'est ce dernier aspect que nous présentons plus avant dans la section qui suit.

4 Programmation et développement par l'utilisateur final

Le terme *programmation par l'utilisateur final*, traduction de *end-user programming*, désigne un ensemble d'outils destiné à des *non-professionnels de l'informatique* pour modifier, étendre, voire créer, des systèmes logiciels. Du point de vue de l'utilisateur, l'objectif premier n'est pas d'apprendre à programmer ou de construire des programmes, mais d'utiliser les outils de programmation comme moyen d'atteindre de nouveaux objectifs. Par exemple, considérant les émissions télévisées, le but est d'enregistrer une émission intéressante, pas de créer un programme d'enregistrement.

Dans ces conditions, la définition d'outils de programmation adaptés à des utilisateurs, *a priori* non motivés par des activités de programmation, pose des difficultés particulières que l'on étudie depuis une bonne quinzaine d'années [Nardi 95, Girard 92, Cypher 93]. L'émergence de l'informatique ambiante redonne à ce domaine de recherche un nouvel élan sous la bannière *end-user development* ou *développement par l'utilisateur final* qui, au-delà du langage de programmation de base, étudie la définition d'outils complémentaires comme l'aide à la mise au point [Lieberman 06], ou la gestion de versions.

Nous explorons ici l'état de l'art en matière de *end-user programming/end-user development* (EUP/EUD) afin d'identifier des pistes pour la conception de méta-IHM. Nous commençons par préciser le problème posé au regard de l'utilisateur, puis nous présentons des solutions selon une classification aujourd'hui bien connue : nous terminons par une synthèse critique de l'offre actuelle dont on tire les leçons pour la définition de méta-IHM pour CONTINUUM.

4.1 Le défi des EUP/EUD

Le défi posé par les EUP/EUD (qui vaut pour tout outil de développement logiciel) revient à établir le juste compromis entre le pouvoir d'expression de l'outil et le coût d'apprentissage pour l'utilisateur. D'après les psychologues, les conditions favorables à l'apprentissage tiennent au bon équilibre entre le défi posé à l'apprenant et son niveau d'expertise [Repenning 06]. Comme le montre la figure 13, un trop grand défi par rapport à l'expertise de l'apprenant provoque anxiété et rejet. Inversement, l'absence de défi peut être génératrice d'ennui.

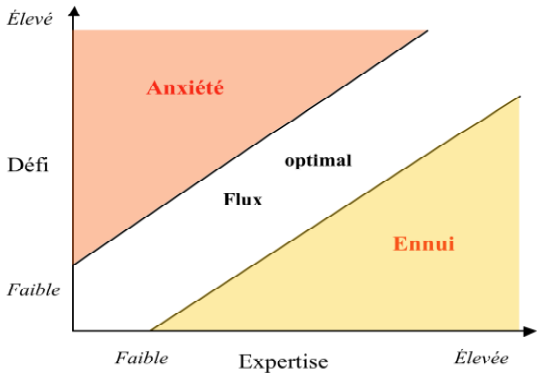


Figure 13. Flux optimal d'apprentissage vu comme le rapport entre défi posé et expertise de l'apprenant. (d'après [Repenning 06])

En reprenant la théorie de Norman, le rapport défi/expertise est optimal si les distances d'exécution et d'évaluation que doit subir l'utilisateur sont conformes à ses compétences et motivations. Les tableurs (spreadsheets) offrent un bon exemple : l'addition d'un ensemble de nombres s'obtient en appliquant l'opérateur SOMME à l'ensemble. Cette technique, qui fonctionne par analogie avec la compétence acquise dès l'école primaire, est facile à assimiler : les distances cognitives sont optimales. Par contraste, en langage impératif comme C, l'algorithme équivalent est loin des formes de pensée courante :

```
sum = 0 ;
for (i=0 ; i<numItems ; i++) {
      sum += items [i] ;
}
return sum ;
```

Mais l'Homme est désireux de progrès. Par utilisation répétée de son EUP/EUD, l'utilisateur est amené à explorer de nouvelles possibilités sous réserve qu'il ne rencontre pas de barrières insurmontables pour son niveau d'expertise. D'où le profil idéal d'EUP/EUD à *pente douce* qu'illustre la figure14.

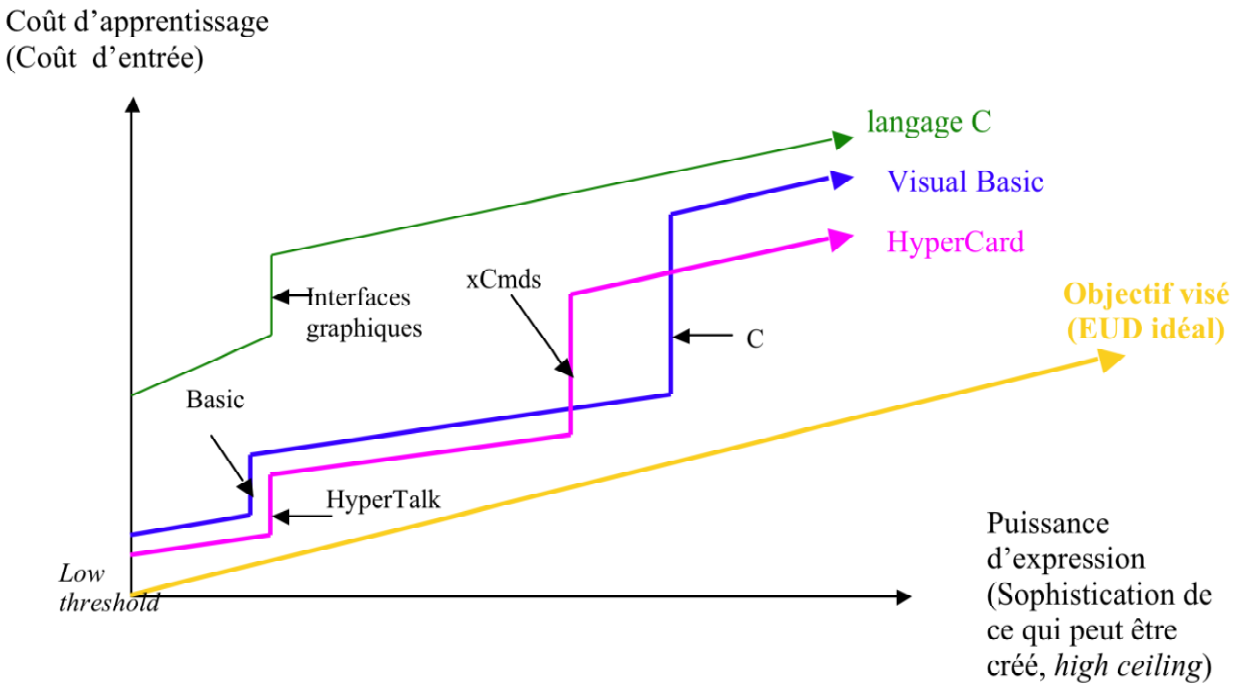


Figure 14. Coût d'apprentissage en fonction de la puissance d'expression d'un EUP/EUD. (d'après [Myers 92])

Par contraste avec le profil de l'EUP/EUD à pente douce et régulière :

- La courbe correspondant au langage C montre que la prise en main du langage est d'emblée difficile, et que, pour programmer des interfaces graphiques, l'utilisateur se heurte à un mur que représente la maîtrise de MFC (Microsoft Foundation Classes).
- Avec Visual Basic, la courbe commence plus bas. Le contact initial est moins violent. Mais la courbe présente deux montées verticales : lorsque l'utilisateur doit apprendre le langage Basic, puis lorsqu'il doit apprendre C.
- Hypercard est simple d'accès, mais pose deux barrières : l'apprentissage du langage HyperTalk, puis celui de xCmds.
- La courbe en pente douce modélise le système idéal des EUP/EUD : un coût d'apprentissage et une puissance d'expression qui progressent régulièrement en cohérence avec le flux optimal

d'apprentissage. Le droit d'entrée est quasi nul (*low threshold*) avec à termes, la capacité de produire des programmes/documents sophistiqués (*high ceiling*).

En synthèse, l'utilisation d'un EUP/EUD doit se voir comme une expérience d'apprentissage. La conséquence, pour le concepteur de tels outils, est que l'utilisateur doit être capable de produire sans erreur des programmes simples pour des objectifs simples, puis, une fois la confiance acquise, l'outil doit lui permettre de découvrir progressivement des fonctions plus puissantes pour atteindre des objectifs complexes[3].

Nous proposons maintenant un diaporama de ces nouveaux systèmes au fil d'une classification proposée par Myers [Myers 00] avant d'en faire la synthèse.

4.2 Langages visuels

Les langages visuels s'appuient sur l'existence, dans nos activités quotidiennes, d'une utilisation massive de représentations graphiques : pictogrammes des panneaux de signalisation routière, icônes de catégorisation des jeux vidéo et DVD, graphiques et dessins dans la presse, voire les nombreuses notations graphiques utilisées en informatique (graphes, state-charts, diagrammes UML, etc.). Si l'utilisation des représentations graphiques est répandue, alors les langages de programmation visuels devraient être plus naturels que les langages classiques. Parmi les nombreuses tentatives fondées sur cette hypothèse, nous retenons : BLOX qui est une ré-écriture du langage Pascal sous forme graphique, AgentSheets inspiré des tableurs, LabVIEW un produit commercial dédié et Fabrik l'un des tous premiers générateurs d'IHM à langage visuel.

4.2.1 Blox

BLOX conserve la sémantique du langage Pascal dont il remplace la syntaxe concrète par des pièces de puzzle [Glinert 87]. L'emboîtement des pièces a le mérite d'interdire les erreurs de syntaxe et met en évidence la structure du programme. La figure 15 montre un exemple de transcription d'un programme textuel dans la syntaxe de BLOX.

4.2.2 AgentSheets

L'objectif d'AgentSheets [Repenning 04] est de permettre à des non-programmeurs de créer des modèles de simulation interactifs hautement parallèles : jeux, histoires, pandémies, etc. L'idée directrice d'AgentSheets est de remplacer les nombres et chaînes de caractères des tableurs par des agents autonomes qui, de ce fait, sont organisés spatialement sur une grille. La contiguïté des cellules exprime la capacité de communication entre les agents qui occupent ces cellules. L'utilisateur crée des agents auxquels il apprend à réagir et à coopérer. Le comportement est exprimé par des règles visuelles (dites règles de réécriture graphiques). Comme pour BLOX, la sémantique du langage est conservée et la syntaxe concrète est remplacée par une notation visuelle. Avec le progrès des techniques d'interaction multimodale, l'utilisateur peut doter les agents de reconnaissance de la parole et d'expression orale (text-to-speech), musicale ou vidéo.

[3] C'est aussi ce que Carroll préconise avec les *training wheels*, par analogie avec les stabilisateurs que l'on installe sur les vélos d'enfant [Carroll 84]. Les stabilisateurs permettent à l'enfant de se familiariser avec la conduite en toute sécurité avant de passer au mode plus défiant sans roulettes.

```
begin
   S1;
   if not I1 then
      begin
         S2;   S3;
      end
   else
      while L1 do
      begin
         if I2 then S4 else S5;   S6;
      end;
   while L2 do S7;
   S8;
end.
```

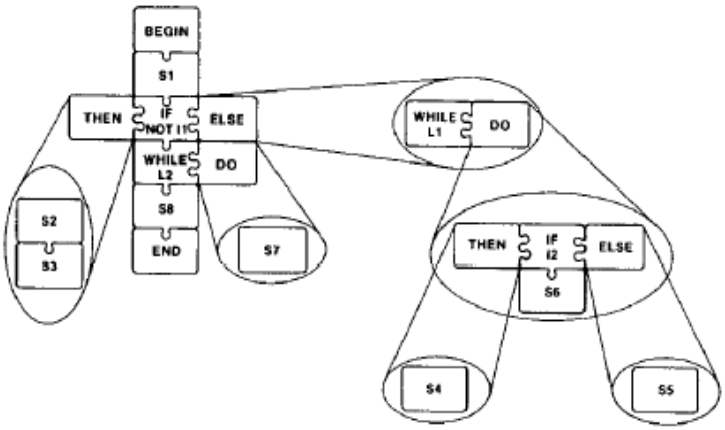


Figure 15. BLOX. En haut, le programme Pascal source. En bas, le même programme dans la syntaxe BLOX. ([Glinert 87])

La figure 16 montre un exemple de simulation de contamination de personnes saines par des malades. En a) la liste des agents (fond, personne saine et personne malade), en b) le comportement qui permet aux personnes de bouger aléatoirement sur le fond et de tomber malade s'il elles rencontrent une personne malade. En c) La simulation résultante.

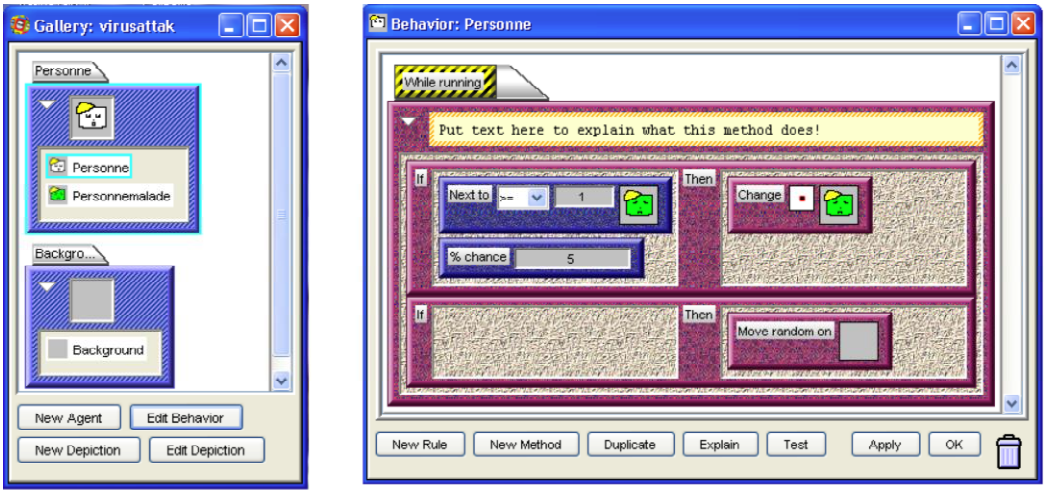
4.2.3 LabVIEW

Créé au milieu des années 80, LabVIEW est aujourd'hui un système commercial [Resendez 03]. Conçu pour la simulation et le contrôle d'appareils de mesure, il est dédié à une catégorie d'utilisateurs bien ciblés. Comme le montre la figure 17, LabVIEW s'appuie sur un modèle de flux de données. À mesure que la connexion des icônes est établie, le programme textuel équivalent se construit.

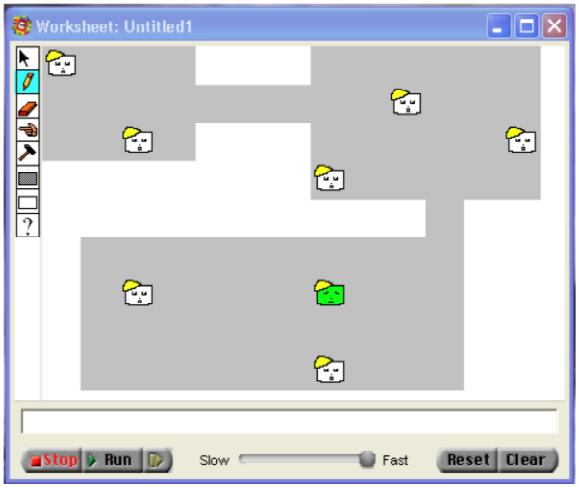
4.2.4 Fabrik

Comme LabVIEW, Fabrik utilise un modèle computationnel de type flux de données. Et comme LabVIEW, il se veut dédié à un type particulier d'activité de programmation : dans le cas de Fabrik, la construction d'IHM graphique. Réalisé en Smalltalk, il est l'un des plus anciens générateurs d'IHM utilisant un langage visuel [Ingalls 88]. S'inspirant de la métaphore des kits de construction, l'environnement propose des briques de base : des objets abstraits (opérateurs arithmétiques) et des objets graphiques (composants d'une interface) que l'utilisateur relie par des connecteurs. Pour définir son application, le concepteur sélectionne les composants désirés à partir d'une bibliothèque, les places dans son espace de travail, et les connecte pour définir leurs fonctionnalités et l'apparence désirée. La Figure 18 donne une idée du kit de construction offert par Fabrik.

Dans la figure 18 a) la fenêtre du haut présente les composants du kit organisés en classes. La fenêtre d'édition du bas contient le programme Fabrik en cours de construction. Dans cet exemple, l'utilisateur a ouvert le tiroir Front Panel et, par drag&drop, a déposé un composant String dans la fenêtre d'édition. En b) un exemple de programme qui démontre la capacité de Fabrik à traiter des flux de données bidirectionnels. Ici, le programme convertit des températures entre Celsius et Fahrenheit. Les températures sont modifiables en déplaçant l'un des curseurs des deux barres verticales.



a) b)



c)

Figure 16. AgentSheets appliqué à la programmation d'un jeu de pandémie. a) la palette des agents (personne saine, personne malade, fond). b) spécification du comportement au moyen de règles Visual AgenTalk. c) le jeu résultat.

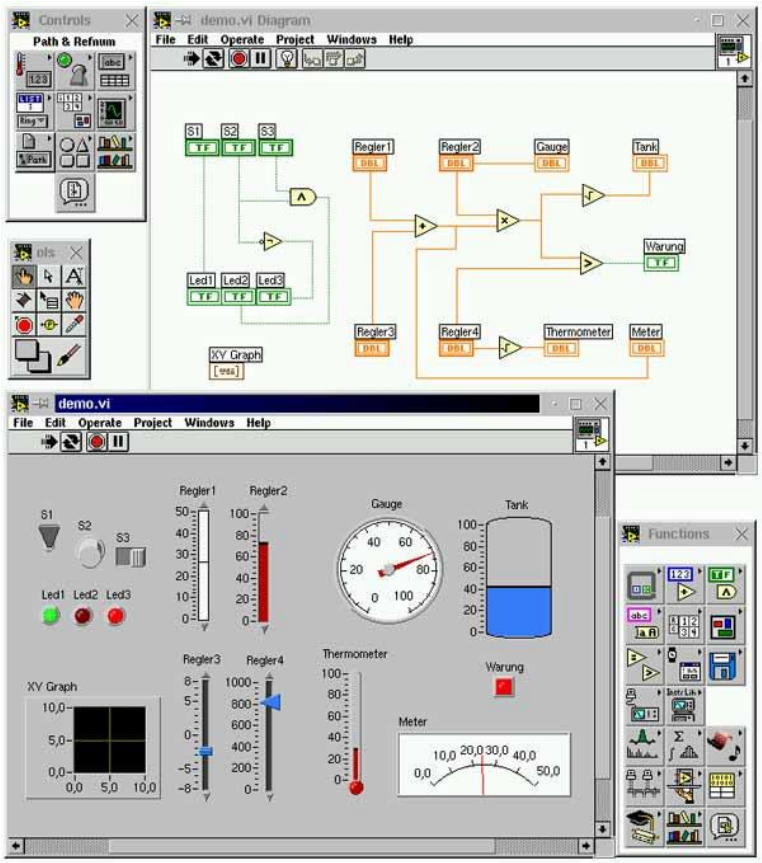


Figure 17. LabVIEW. ([Resendez 03])

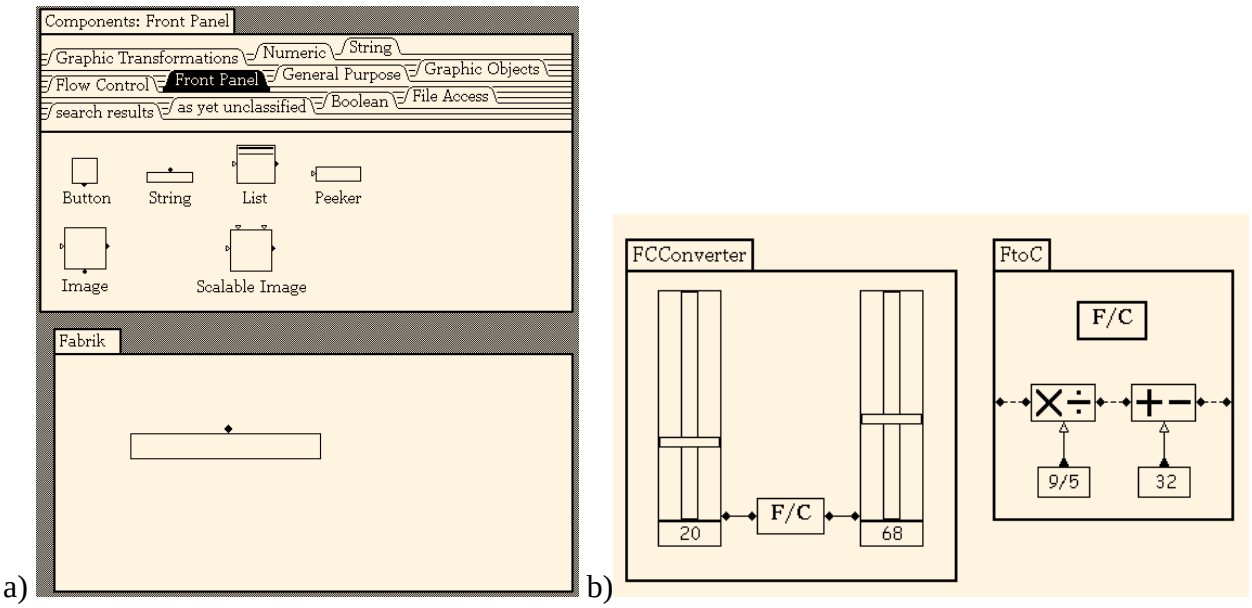


Figure 18. (a) Fabrik et son atelier. (b) Un exemple de programme avec gestion de flux bidirectionnel. ([Ingalls 88])

4.2.5 Synthèse sur les langages graphiques

Quelques langages visuels comme LabVIEW rencontrent un véritable succès parce que leur domaine de couverture est ciblé. Mais, de manière générale, les langages visuels ne sont pas la panacée. On trouvera dans [Whitley 97] une synthèse de nombreux résultats de psychologie expérimentale : en somme, les notations graphiques sont adaptées à des tâches ou domaines précis, mais ne peuvent pas prétendre être

universellement supérieures pour tout EUP/EUD. D'où le retour vers la programmation textuelle sous forme de scripts.

4.3 Langages textuels de script

La programmation textuelle pour utilisateur final doit viser concision et richesse d'expression dans le respect du rapport défi/expertise. L'approche la plus fréquemment retenue est l'intégration de services de programmation textuelle dans des applications existantes. Il peut s'agir de langages textuels dédiés comme pour l'expression des filtres dans le courrier électronique jusqu'à l'inclusion de langages pour professionnels comme LISP dans EMACS.

À l'origine, les langages de script visaient à automatiser des séquences d'actions utilisateur et à les encapsuler en une unité exécutable à la demande. Les macro-commandes relèvent de cette tendance (commandes du Shell Unix, macros dans Word, etc.). Chickenfoot et HyperTalk présentés ci-dessous en sont d'autres illustrations. Avec le temps, les langages de script sont devenus des langages pour professionnels appréciés pour leur caractère interprété (JavaScript, Tcl, Python, Perl et bien d'autres). Nous présentons ci-dessous le langage Logo de Papert, exemple séminal en matière de programmation textuelle pour non-experts, suivi de HyperTalk et Chickenfoot.

4.3.1 Logo

Logo a été conçu dès 1967 par Papert[4] pour démystifier la programmation des ordinateurs [Solomon 78]. Selon Papert, Logo se caractérise par « un seuil faible [de prise en mains] et l'absence de plafond ». En d'autres termes, Logo est censé être un langage à « pente douce ». Il se veut accessible aux novices mais aussi aux experts pour programmer des problèmes de simulations et la création de présentations multimédia. En figure 19, l'un des plus célèbres environnements Logo : la Tortue (*Turtle*). Une interface de commande permet à l'utilisateur d'écrire les ordres qu'un robot-tortue doit exécuter. Il convient de noter que Logo a inspiré de nombreux jouets-robots programmables pour enfants. Voir aussi les célèbres successeurs commerciaux de Logo dont les Lego Mindstorms [Papert 80].

4.3.2 HyperCard

HyperCard livré dès la sortie des premiers Macintosh, est un environnement pionnier pour la production, par des non-professionnels, de documents hypermédia. L'innovation tient à l'intégration du mode *auteur* dans le document qui permet d'éditer et de programmer de nouvelles cartes au moyen d'un langage de script : HyperTalk. Comme le montre l'exemple suivant, HyperTalk s'appuie sur un modèle à événement :

```
on mouseUp
  put "100,100" into pos
    repeat with x = 1 to the number of card buttons
      set the location of card button x to pos
      add 15 to item 1 of pos
  end repeat
end mouseUp
```

[4] Papert est un mathématicien. À son retour de Genève où il a travaillé avec Piaget, il fonde avec Minsky le laboratoire d'Intelligence Artificielle au MIT.

4.3.3 Chickenfoot

Chickenfoot [Bolin 05], un outil de programmation intégrable dans le browser Firefox, permet d'écrire des scripts pour manipuler des pages Web et automatiser des recherches sur le Web sans « voir » le langage html. Chickenfoot est un surensemble de Javascript qui inclut des fonctions spéciales pour le Web. Une fois installé, l'éditeur est accessible via la barre de menu de FireFox. L'exemple de la figure 20 montre comment l'utilisateur peut automatiser une action de recherche de page web. En a) l'accès à l'éditeur Chickenfoot et en b) le code écrit est encadré en rouge, il permet de faire une recherche plus rapidement.

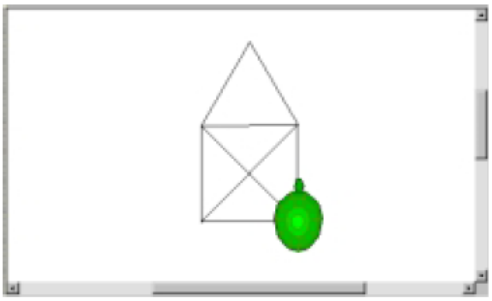
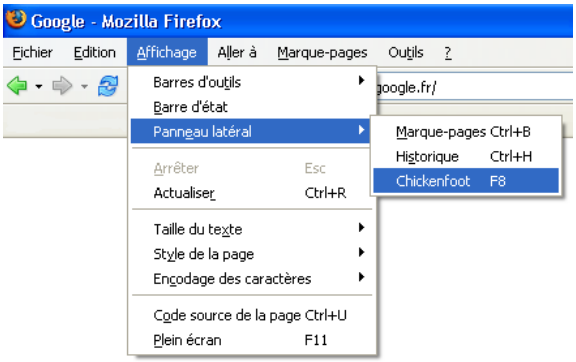
Instructions	Résultat
AVANCE *100* DROITE *30* AVANCE *100* DROITE *120* AVANCE *100* DROITE *120* AVANCE *100* GAUCHE *135* AVANCE *141.42* GAUCHE *135* AVANCE *100* GAUCHE *135* AVANCE *141.42* GAUCHE *135* AVANCE *100* GAUCHE *90*	

Figure 19. L'environnement Logo et l'exemple de la Tortue. ([Solomon 78])

4.3.4 Synthèse sur les langages de script

A l'exception de Logo, les langages de script s'apparentent aux langages de programmation usuels (itérations, traitant d'événements). Mackay, dans une étude un peu ancienne [Mackay 90], constate que la création de scripts est avant tout le fait des techniciens, que ces scripts se disséminent, personnalisés ensuite par les non-experts. La prise en compte de la dimension « social programming » est explicite dans la conception de Scratch, un langage de script textuel augmenté de sucre syntaxique graphique [Resnik 09] (cf. Figure 21). L'approche par script qui semble la mieux admise par les non-experts est la construction de macros qui encapsulent des actions utilisateur en un tout ré-utilisable. D'où l'exploration de la piste « programmation par l'exemple ».

a)



b)

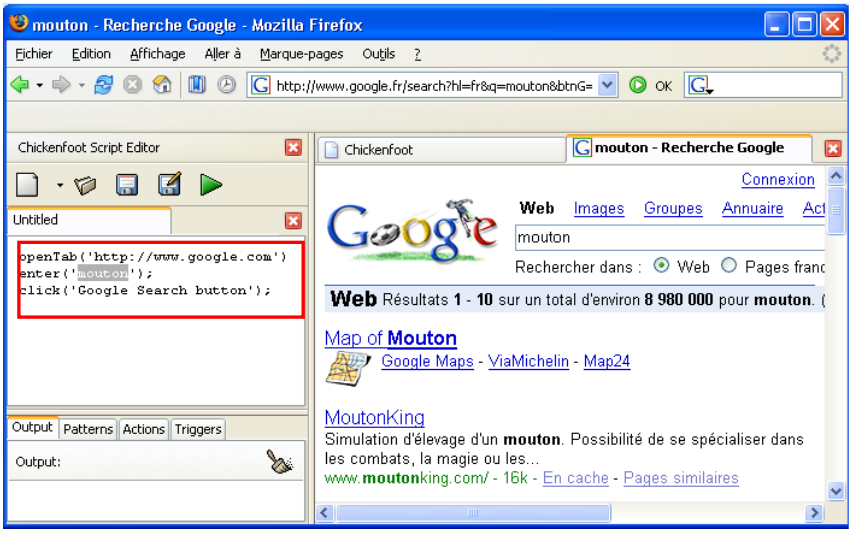


Figure 20. a) l'accès à Chickenfoot b) un exemple de simplification de recherche avec le code encadré en rouge.

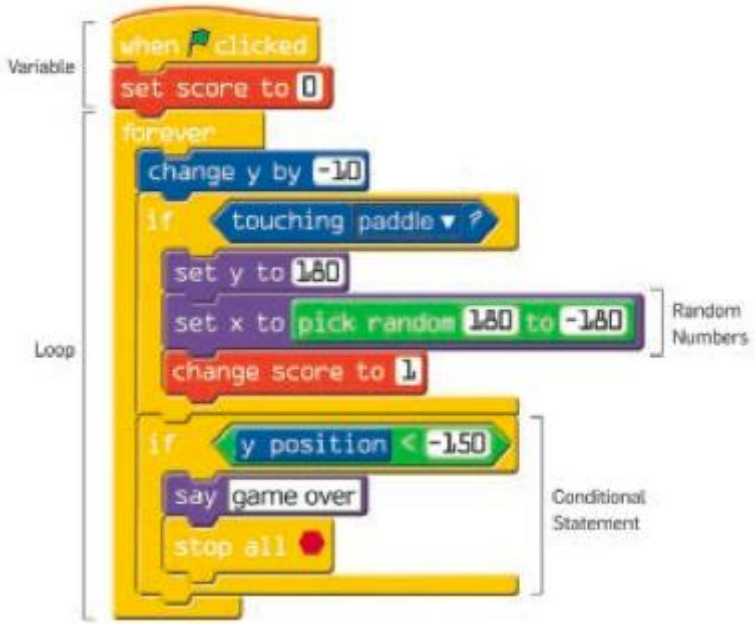


Figure 21. Un exemple de script Scratch.

4.4 Programmation par l'exemple

La programmation par l'exemple est une technique pour enseigner à l'ordinateur de nouveaux comportements en lui montrant des actions sur des exemples concrets. Le système observe les actions de l'utilisateur, repère les paramètres, et généralise le tout sous forme d'un programme. *Fait ce que j'ai fait*[5] est une manière concise de caractériser ce type de système. Si le système est capable d'inférence, il peut aussi deviner les intentions de l'utilisateur. *Fait ce que je veux dire*[6] qualifie alors ce type de système. Pour illustrer ces différences, nous proposons quelques exemples pour leur caractère pionnier appliqués à des domaines distincts : Emacs (pour le traitement de texte), Eager (pour la détection de tâches répétitives) et Peridot (pour la conception et la génération d'IHM graphique). Mais d'autres exemples méritent également d'être relevés, notamment les travaux de Lieberman sur Grammex (pour la définition de grammaires) ou Mondrian (pour l'extraction de connaissances expertes) [Lieberman 01].

4.4.1 Emacs

Emacs est un traitement de texte qui tente pour la première fois d'intégrer la programmation par l'exemple. Lorsque l'utilisateur entre en mode « macro », Emacs enregistre et interprète au fur et à mesure les commandes de l'utilisateur. Ce dernier peut ainsi évaluer directement les effets de la macro qui pourra être appliquée ultérieurement à d'autres composantes de texte sur demande explicite de l'utilisateur. La création et manipulation de macros sont accessibles via des commandes prédéfinies :

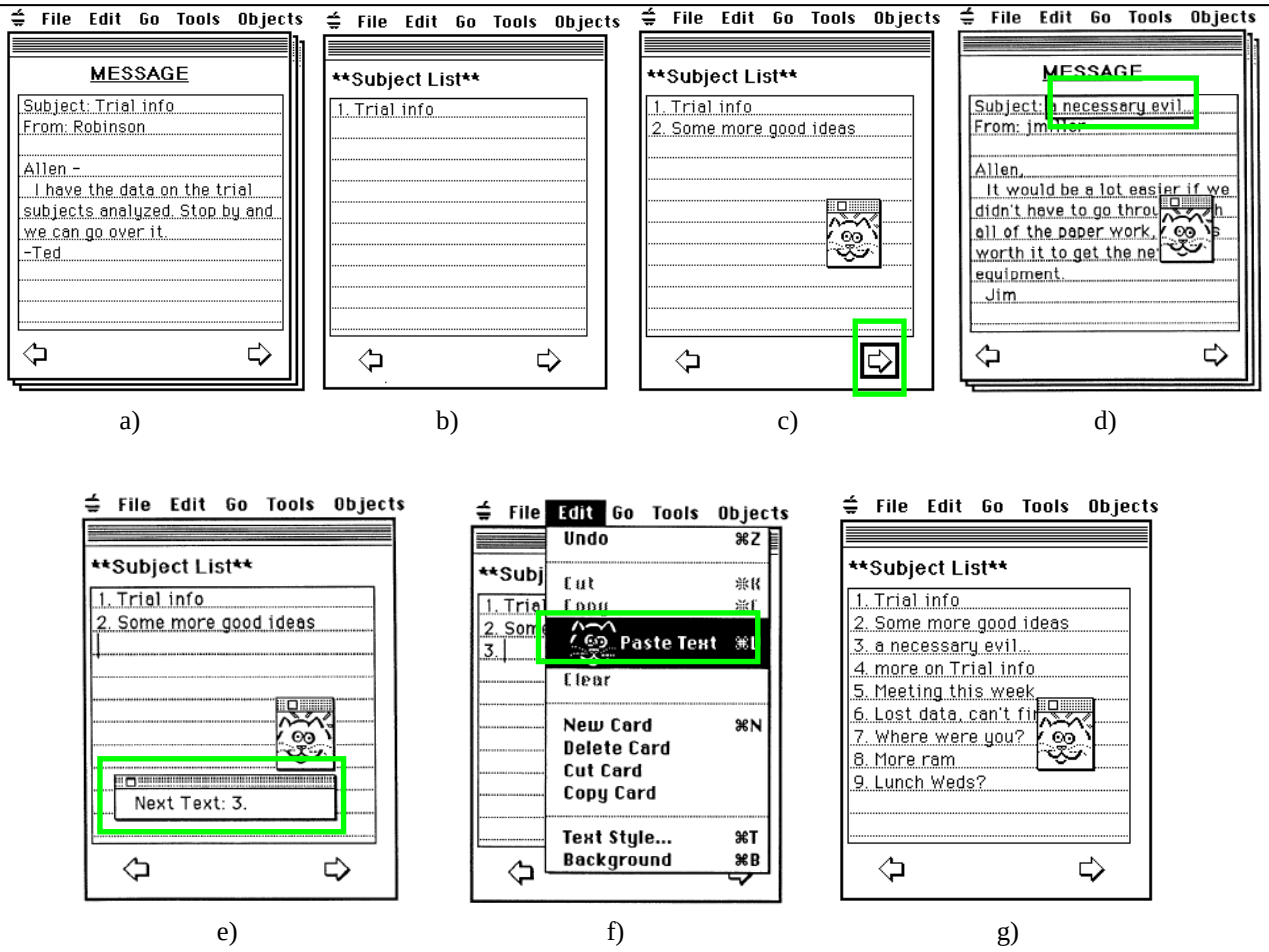
- C-x (: (start-kbd-macro) Commence la définition d'une macro
- C-x) : (end-kbd-macro) Finit la définition d'une macro
- C-x e : (call-last-kbd-macro) Exécute la dernière macro définie
- M-n C-x e : Exécute n fois la dernière macro définie
- C-u C-x (: Exécute la dernière macro définie, puis rajoute à la fin de la macro les commandes tapées jusqu'à la rencontre de la commande (end-kbd-macro)
- name-last-kbd-macro : nomme la dernière macro définie (essentiel pour la sauver)
- insert-kbd-macro : insère une macro dans un fichier
- load-file : charge un fichier contenant une macro
- M-x nom de la macro : exécute la macro nom de la macro.

Contrairement à Eager, Emacs n'inclut pas de mécanismes d'inférence.

4.4.2 Eager

Eager [Cypher 91], développé pour HyperCard, détecte les tâches répétitives par inférence. Lorsque l'utilisateur effectue deux fois de suite la même action sémantique mais sur des données distinctes, l'icône représentative de Eager (un chat) surgit à l'écran et les actions que l'utilisateur est censé appliquer, sont affichées en vert. Si l'utilisateur effectue les actions pressenties par Eager, alors Eager propose de prendre en charge la suite de la tâche. La figure 22 montre l'exemple d'un utilisateur qui souhaite constituer un document à partir du champ « sujet » de tous les messages électroniques reçus. Comme Eager, Metamouse permet de détecter des taches répétitives ([Maulsby 89]).

[5] En anglais, *do what I do.*
[6] En anglais, *do what I mean.*



Un utilisateur veut constituer une liste des sujets de ses mails. **a)** Il copie le sujet du premier message et **b)** le recopie dans la liste en première position. Il fait de même pour le second sujet. **c)** Alors, l'icône d'Eager apparaît : le système a détecté un patron dans les actions de l'utilisateur. Eager encadre la flèche de droite en vert, anticipant le fait que l'utilisateur va la cliquer. En effet, l'utilisateur sélectionne la flèche. Eager anticipe les actions suivantes: **d)** sélection du sujet, **e)** numérotation du sujet, **f)** recopie du sujet dans la liste. L'utilisateur, maintenant confiant dans les prédictions de Eager, **g)** clique l'icône d'Eager pour que le système automatise la tâche et la termine.

FiFigure 22. Eager en action. (d'après [Cypher 91])

4.4.3 Peridot

Bien qu'ancien, Péridot reste la référence en matière de génération d'IHM par l'exemple [Myers 86]. Peridot crée automatiquement le code de l'interface tandis que l'utilisateur démontre ce à quoi l'interface doit ressembler, comment elle fonctionne et comment l'utilisateur final interagit avec elle. Le concepteur dessine des images de la future IHM et définit les actions que l'utilisateur pourra effectuer (déplacer la souris, sélectionner un composant, saisir du texte, etc.). Pour simuler les actions de l'utilisateur final, le concepteur dispose d'icônes de simulation représentant les actions possibles. Pour chaque action du concepteur, le système cherche à deviner par inférence les relations entre l'action, les éléments présents de l'interface et le contexte. Avant toute décision, Peridot demande au concepteur une confirmation de la solution présumée. Si la solution est incorrecte, d'autres solutions sont proposées ou bien le concepteur peut spécifier explicitement la solution dans un langage textuel.

4.4.4 Synthèse sur la programmation par l'exemple

La programmation par l'exemple semble séduisante puisqu'elle n'impose pas à l'utilisateur d'apprendre un nouveau langage. De plus, lorsque le mode « programmation » est intégré à l'application comme dans Emacs et Eager, l'utilisateur final programme sans le savoir. A contrario, pour les systèmes munis de mécanismes d'inférence, il convient de fournir de bons exemples. Or, trouver les bons exemples implique, pour l'utilisateur, de comprendre le fonctionnement du système. (On connaît la difficulté de fournir à Word de bons exemples pour la mise en page automatique.) On retombe alors sur le problème usuel des distances d'exécution et d'évaluation que l'utilisateur doit franchir pour atteindre ses objectifs.

Ainsi, programmation visuelle, scripting textuel et programmation par l'exemple ont chacun leur raison d'être et leurs difficultés. D'où l'émergence progressive d'outils de programmation mixtes comme dans les environnements auteur.

4.5 Environnements auteur

L'élément distinctif et innovant des environnements auteur est leur façon d'intégrer les services de programmation avec la présentation du produit final. Ce produit peut être un document hypermédia sophistiqué (*high ceiling*) mais aussi un document simple demandant peu d'effort d'apprentissage (*low threshold*). Nous avons retenu StageCast comme environnement destiné aux enfants, DreamWeaver ou Flash comme outils commerciaux, Alice comme exemple phare académique et Whyline comme nouveau concept pour la mise au point.

4.5.1 Stagecast

Stagecast [Smith 94], aussi connu par ses anciens noms Cocoa ou KidSim, est un environnement qui permet aux enfants de créer des simulations par la combinaison de programmation visuelle, textuelle et par l'exemple. La figure 23 montre la création d'une simulation. L'enfant déplace les objets de la zone c) sur la zone de simulation a). En b) il peut contrôler la zone de temps. Pour mettre en œuvre sa simulation, il crée des règles de comportement en d). Chaque règle peut être éditée en e) par une programmation textuelle ou par l'exemple. Dans ce cas, l'enfant a programmé une règle grâce à l'exemple : lorsque le gorille se retrouve au bord d'une branche il en descendra.

4.5.2 DreamWeaver

DreamWeaver est un outil pour la conception de site web. Le concepteur débutant peut élaborer ses pages web par manipulation directe de textes, d'images ou d'animations. Le code html est généré au fur à mesure. Une fois familier avec l'outil, l'utilisateur a le choix entre la visualisation graphique et la visualisation textuelle. S'il modifie le code textuel, la version graphique est modifiée en cohérence et vice versa. DreamWeaver applique donc le principe d'égale opportunité. La figure 24 montre en a) la partie visuelle, et en b) la partie textuelle. Comme DreamWeaver, Flash offre une double manipulation de l'objet créé.

4.5.3 Alice

Alice [Cooper 00] est un outil de création de scènes animées de réalité virtuelle en 3D. Cet environnement est avant tout destiné à des non spécialistes de l'informatique. L'utilisateur débutant commence par une interface de programmation visuelle très simple. Plus il avance, plus il peut raffiner sa programmation jusqu'à créer des procédures de complexité croissante. Le logiciel permet de raconter des histoires, de créer des personnages, des décors, à partir d'une riche galerie d'objets. La figure 25 montre l'interface générale de

Alice. En a) la bibliothèque d'objets, en b) le monde 3D en création, en c) la liste des événements, en d) les informations sur l'objet sélectionné, et en e) la zone de code.

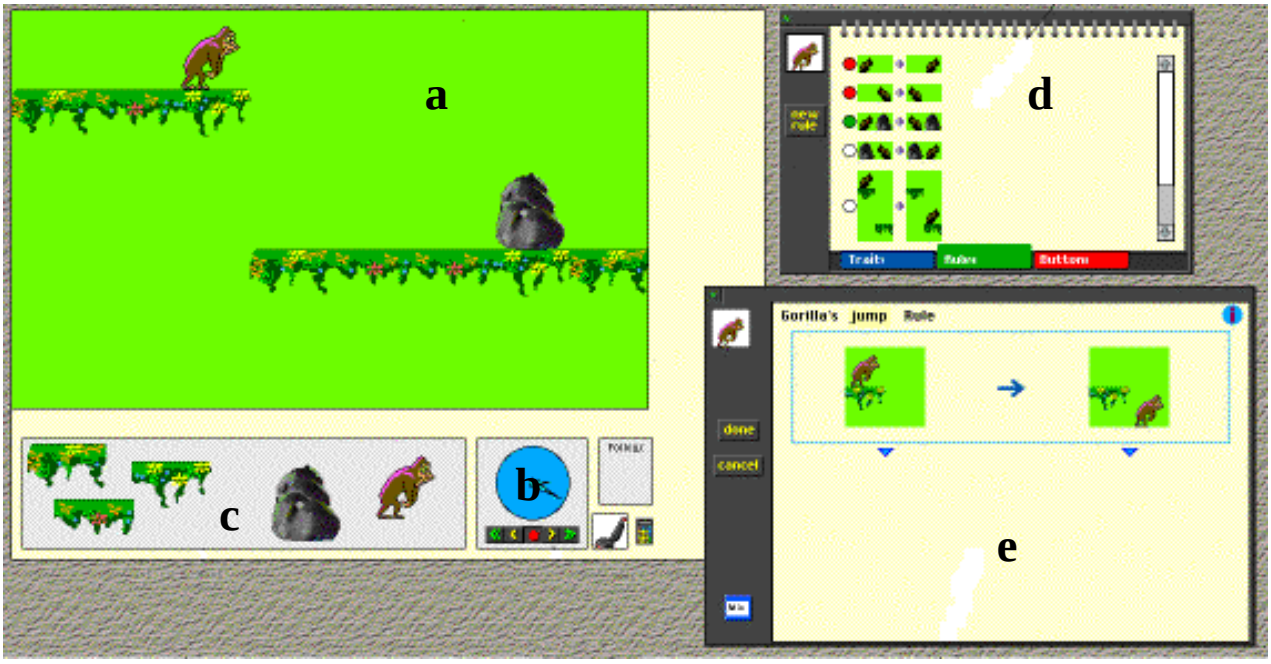
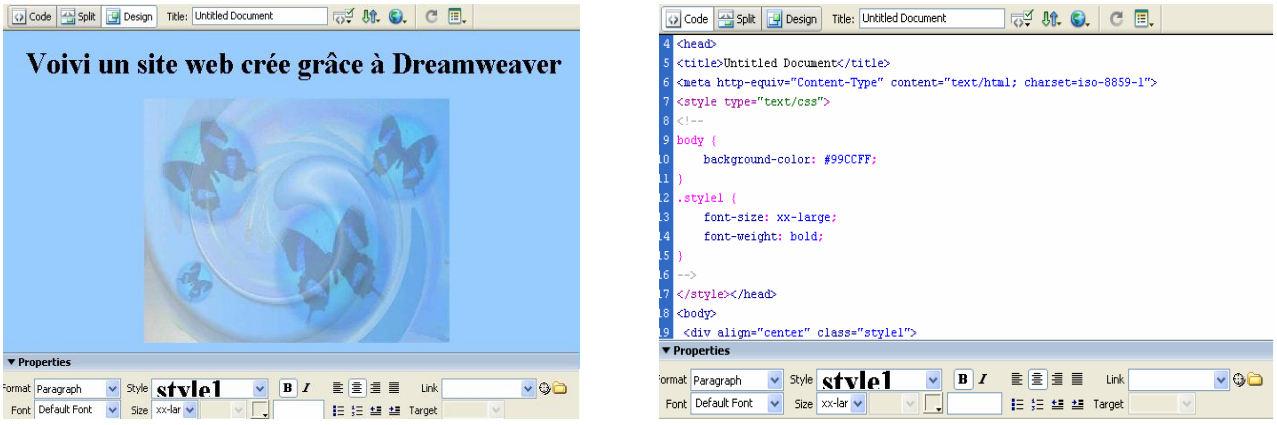


Figure 23. StageCast. a) la simulation b) la zone de temps c) les objets déplaçables d) les règles de comportement e) Editions d'une règle de comportement. ([Smith 94]).



a) b)

Figure 24. DreamWeaver. a) la construction d'une page de manière visuelle. b) l'équivalent en mode textuel.

Comprendre le code et trouver les erreurs d'un programme sont des activités essentielles en développement de logiciel. Curieusement, les environnements auteurs comme Flash, qui sont destinés à des novices, proposent les mêmes techniques de mise au point que l'on pratique depuis une trentaine d'années : points d'arrêt, affichage de valeurs de variables, etc. Whyline apporte quelques éléments de réponse à ce problème.

4.5.4 Whyline

Whyline [Myers 06] offre une nouvelle façon de mettre au point un programme en répondant à des questions de type « pourquoi ». Après enquête auprès d'utilisateurs novices, il se trouve que les développeurs, en phase de mise au point, se posent souvent les questions *pourquoi ?* et *pourquoi pas ?* Whyline répertorie les dernières actions de l'utilisateur et propose via un menu une liste de questions.

Lorsque l'utilisateur sélectionne une question, le système lui répond. L'utilisateur peut ainsi comprendre et déboguer une action qu'il a mal exécutée ou que le système a produite. La figure 26 montre l'exemple d'un tel menu pour une application de traitement de texte.

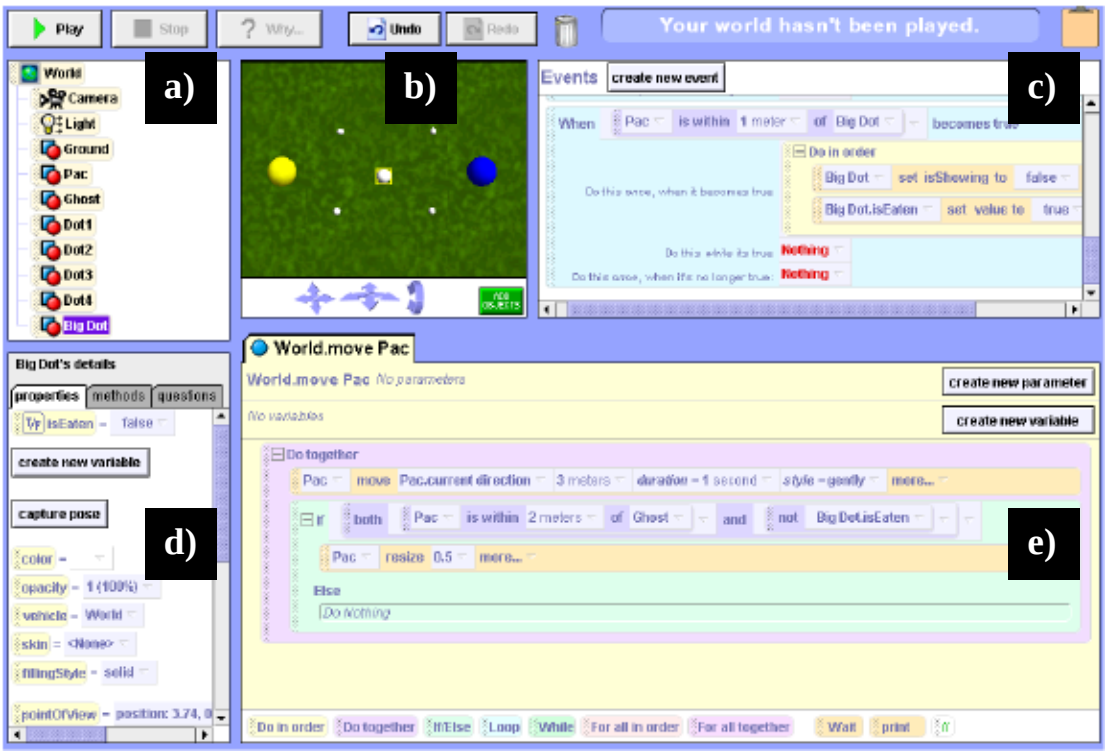


Figure 25. Alice. a) la bibliothèque d'objets, b) le monde 3D en création, c) la liste d'événements, d) les informations sur l'objet sélectionné, e) la zone de code. ([Cooper 00])

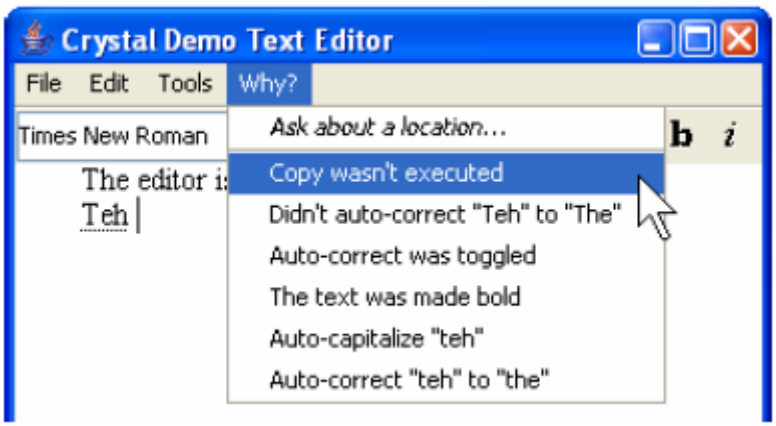


Figure 26. Le concept de Whyline. ([Myers 06])

4.6 Synthèse

Pour conclure, aucun exemple de l'état de l'art en matière de EUP/EUD n'a démontré sa capacité à satisfaire le double requis *low threshold, high ceiling* sans devoir franchir de barrières cognitives. Aucune notation (graphique ou textuelle), aucune approche (par démonstration ou non) ne peut à elle seule satisfaire ces requis. Enfin, programmer, même à petite échelle, n'est pas simplement produire du code, c'est aussi le mettre au point. De nouveaux outils de mise au point adaptés à l'utilisateur final doivent être inventés.

Dans la section chapitre qui suit, nous exploitons les idées du concept « End-user programming » ainsi que les résultats de notre taxonomie pour la conception de méta-IHM pour CONTINUUM.

5 Méta-IHM et points de contrôle utilisateur dans l'architecture CONTINUUM

CONTINUUM intègre dans sa conception architecturale la capacité pour l'utilisateur de contrôler ses espaces ambiants. Nous proposons pour cela le concept de « point de contrôle utilisateur ».

Nous convenons d'appeler *point de contrôle utilisateur*, un point d'intervention de l'utilisateur dans le processus d'adaptation du système en réaction aux changements de contexte. Du point de vue technique, l'emplacement des points de contrôle dépend de la décomposition fonctionnelle du système. La décomposition fonctionnelle de la plate-forme CONTINUUM, qui procède par analogie avec le modèle des comportements cognitifs de Rasmussen, comprend 3 niveaux d'adaptation (pour plus de détail, se reporter au livrable D2) :

- *Adaptation comportementale réflexe* : à un contexte connu, correspond un plan d'adaptation directement applicable. Concrètement, dans le cas de CONTINUUM, ce plan correspond à un ensemble d'Aspects d'Assemblage (AA) déployés c.-à-d. présents dans le AA Designer de l'application et donc directement applicables à un assemblage de composants source (qui se trouve être l'application à adapter à la situation).
- *Adaptation contextuelle tactique* : à un contexte connu, correspond plusieurs plans applicables ; il suffit de choisir le plan le plus adapté, c'est-à-dire celui qui correspond à la situation courante. Concrètement, dans le cas de CONTINUUM, il s'agit pour le Gestionnaire de Contexte (GC) d'interroger la Base de Connaissances (BdC) pour identifier les prédicats vérifiés et de là, identifier la situation actuelle et déployer les AA attachés à cette situation dans le AA Designer de l'application. Nous retombons ensuite sur les traitements de l'adaptation réflexe.
- *Adaptation stratégique* à des contextes inconnus : la solution doit être construite. Comme indiqué dans le livrable D2, le GC et la BdC ne seront pas enrichis par des algorithmes d'apprentissage automatiques. Nous convenons que l'enrichissement se fera par l'utilisateur complété plus tard par des capacités système d'apprentissage.

Puisqu'il s'agit pour l'utilisateur d'intervenir dans le processus d'adaptation, nous définissons *a minima* un point de contrôle par niveau d'adaptation. À chacun de ces points de contrôle correspond une méta-IHM. La figure 26, tirée du livrable D2, rend explicite la présence des points de contrôle dans l'architecture CONTINUUM :

- *Une méta-IHM-réflexe* dont l'exécution a un impact direct sur le AA Designer de l'application courante.
- *Une méta-IHM-tactique-BdC* pour explorer et éditer la BdC du niveau tactique.
- *Une méta-IHM-tactique-GC* pour explorer et modifier le GC du niveau tactique.
- *Une méta-IHM-stratégique* pour intervenir dans le processus d'apprentissage de nouvelles situations.

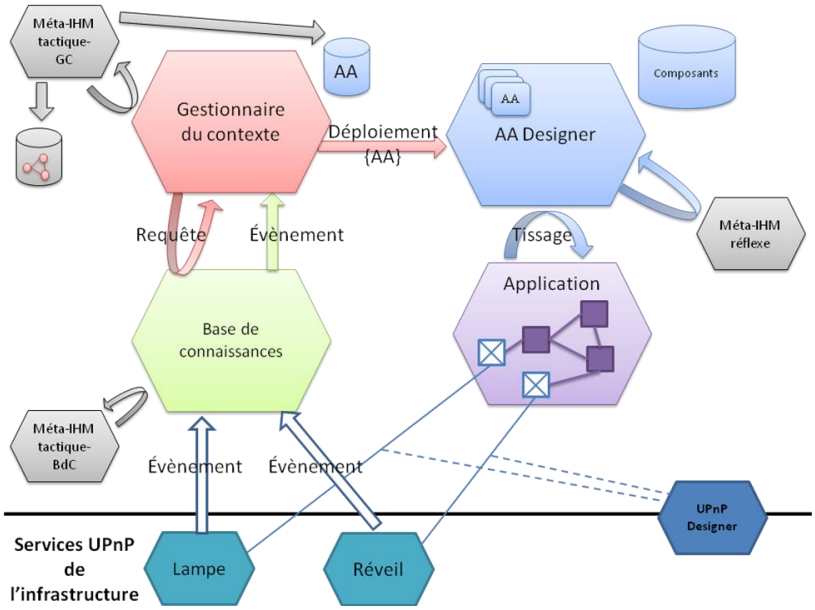


Figure 26. Les points de contrôle de l'architecture CONTINUUM.

Dans le temps imparti au projet CONTINUUM, nous convenons de concentrer nos efforts comme suit :

- *Méta-IHM-réflexe* : doit permettre à l'utilisateur d'influer sur l'adaptation du système à la volée (par exemple, redistribuer l'IHM de son application avec l'arrivée de nouvelles ressources d'interaction), d'intervenir éventuellement sur le choix des AA à exécuter. Ce dernier point est à vérifier via le développement des démonstrateurs. À titre indicatif, un exemple de méta-IHM-réflexe est fourni dans la section suivante où l'utilisateur Bob du scénario prospectif demande une redistribution de l'IHM de l'application Photo-Browser.
- *Méta-IHM-tactique-BdC* : dans un premier temps, cette méta-IHM sera minimaliste. Elle ne fera pas l'objet d'une évaluation centrée utilisateur final, mais suffira aux besoins de mise au point des démonstrateurs CONTINUUM.
- *Une méta-IHM-tactique-GC* : Il s'agira essentiellement de « programmer » en amont (ou à la volée) les comportements que le système devra respecter pour des situations considérées comme clefs par les utilisateurs et de là, mettre au point des programmes, les retrouver, les éditer. Cette méta-IHM est au cœur de notre contribution de recherche.
- *Une méta-IHM-stratégique* pour intervenir dans le processus d'apprentissage de nouvelles situations. Notre hypothèse est la possibilité de réutiliser à la volée les méta-IHM du niveau tactique. Ce point est à vérifier avec le développement des démonstrateurs.

La section qui suit décrit nos premières conceptions de méta-IHM pour les niveaux réflexe et tactique.

6 Premières propositions de Méta-IHM pour CONTINUUM

6.1 Méta-IHM-réflexe

La figure 27 montre un exemple de méta-IHM-réflexe fondée sur le geste pour contrôler l'utilisation de l'application Photo-Browser dans un espace ambiant. Cette application correspond à la séquence « Alice et Bob au restaurant » du scénario prospectif de CONTINUUM. Il s'agit d'une méta-IHM purement exploratoire motivée par des préoccupations techniques de faisabilité d'intégration de Photo-Browser sur la plate-forme CONTINUUM. Dans l'implémentation actuelle, le geste est interprété par la technique du Magicien d'Oz (c.-à-d., la reconnaissance du geste et de sa sémantique est assurée par un compère humain).

Comme le montre la figure 27, l'application est lancée en mettant le smartphone en contact avec la table : alors, l'utilisateur peut trier ses photos sur la table via l'IHM graphique à plusieurs mains de Photo-Browser. Cette IHM est rendue possible par les propriétés techniques de la table MERL dite multi-points. L'IHM de l'application est alors centralisée. L'utilisateur peut demander la répartition de l'IHM de Photo-Browser entre la table et le mur (géré par un PC) par un geste de balayage à deux mains allant de la table vers le mur. Ce geste est reconnu par la méta-IHM de l'espace ambiant : la photo qui était sélectionnée sur la table se trouve maintenant affichée à la fois sur le mur et la table. Toute nouvelle sélection de photo sur la table est actualisée sur le mur. Les ressources d'interaction de l'espace ambiant comprennent la table et le mur. L'utilisateur peut y ajouter celles du Smartphone en le posant sur la surface de la table. Ce geste, reconnu par la méta-IHM de l'espace ambiant, permet d'utiliser le smartphone comme télécommande : le smartphone permet de naviguer séquentiellement dans l'espace des photos grâce aux boutons « next » and « previous » affichés sur son écran : les photos sont affichées de manière synchronisée à la fois sur le mur et la table. L'IHM de PhotoBrowser est maintenant répartie sur la table, le mur et le smartphone. La méta-IHM de l'espace ambiant permet de supprimer la table de l'espace ambiant par un balayage de la main sur sa surface.

Dans notre espace taxonomique de la Figure 3, la méta-IHM de l'espace ambiant de la figure 27 se caractérise ainsi :

- Nature des entités manipulées : 2 entités mixtes intrinsèquement numériques (la table MERL et le gPhone) ; 1 entité mixte intrinsèquement physique (le mur couplé à un un PC et son vidéoprojecteur).
- Type de manipulation : directe (l'utilisateur désigne du geste les entités table et mur et met le smartphone en contact avec la table).
- Services offerts : découverte (le système découvre et gère l'arrivée et le départ dynamique des entités manipulées – table, smartphone, mur), assemblage dynamique (de la table, du smartphone et du mur), redistribution d'IHM (entre la table, le mur et le smartphhone), mais absence de paramétrage et de remodelage.

Méta-UI-réflexe (à gauche): Lancement de Photo-Browser en mettant le smartphone en contact avec la table
Application photo-Browser (centre): Tri de photos sur la table
Méta-UI-réflexe (à droite): Arrêt de la table par un geste de balayage couvrant la table

Méta-UI-réflexe (ci-dessus) : Geste de balayage des 2 mains depuis la table vers le mur pour redistribuer l'IHM de Photo-Browser : la photo actuellement sélectionnée sur la table est également visualisée dans le navigateur web attaché au mur ; l'utilisateur peut naviguer dans l'espace des photos via ce navigateur et via la table.

Méta-UI-réflexe (ci-contre) : Pose du smartphone sur la table pour en demander lecouplage ; le smartphone sert alors de télécommande aussi bien pour la table que pour le mur pour naviguer séquentiellement dans l'espace des photos.

Figure 27. Exemple de Méta-IHM-réflexe pour Photo-Browser.

- Niveau de contrôle des services : contrôlabilité des 3 services offerts (découverte, assemblage et redistribution), mais pas d'observabilité ni de traçabilité : rien n'indique à l'utilisateur qu'il peut coupler le smartphone à la table ni qu'il peut redistribuer l'IHM de l'application.
- L'IHM de cette méta-IHM est externe : elle pourrait être utilisée pour une autre application dont l'IHM graphique peut être redistribuée.
- Extensibilité du langage d'interaction : aucune.

Considérons les points de contrôle explicités dans la figure 26 avec l'hypothèse que la séquence du scénarimage de la figure 27 correspond à une seule situation. Alors, l'interprétation des gestes utilisateur (par l'IHM de la méta-IHM) donne lieu à la génération d'une commande (qui fait sens pour la méta-IHM : en l'occurrence, découvrir, assembler, redistribuer). La méta-IHM traduit cette commande en le choix du (ou des) AA idoine(s) déjà déployés dans le AA designer de Photo-Browser.

Dans l'hypothèse où la séquence de la figure 27 correspond à plusieurs situations, les gestes utilisateur donnent lieu à des modifications de la Base de Connaissances (BdC). La BdC en avertit le gestionnaire de contexte qui est alors en mesure d'identifier la situation et de déployer l'ensemble correspondant de AA dans le AA designer de l'application Photo-Browser.

6.2 Méta-IHM-tactique pour la Base de Connaissances

La figure 28 montre l'IHM web actuelle de la méta-IHM pour explorer et éditer la base de connaissances. La sélection de l'onglet « Show Context » permet d'obtenir une représentation graphique des informations contextuelles de la BdC. « Edit Context » permet de la modifier, etc.

Si le temps le permet, des améliorations tirées de l'état de l'art sur l'exploration et l'édition interactives de grands espaces d'information seront apportées à cette version qui a le mérite d'être opérationnelle.

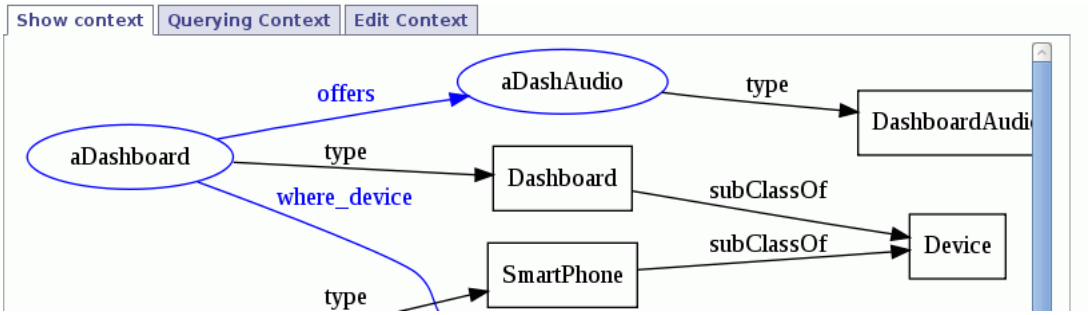


Figure 28. Représentation Graphique des informations contextuelles dans la BdC.

6.3 Méta-IHM-tactique pour la Gestion du Contexte

Cette méta-IHM est au cœur de notre recherche IHM dans CONTINUUM. Il s'agit de fournir à l'utilisateur final un environnement de développement lui permettant de contrôler son espace ambiant. Dans le temps imparti au projet, on accordera la priorité à la définition du langage. Ce travail de conception est en cours : la conception est partielle et n'a pas fait l'objet d'évaluation formelle avec des utilisateurs représentatifs. En l'état, notre conception s'appuie sur notre analyse du problème, sur des tests informels avec 4 utilisateurs ainsi que sur nos connaissances expertes en Interaction Homme-Machine. Notre proposition est donc perfectible et sera modifiée par itération.

6.3.1 Principes

Comme première étape, nous faisons les choix suivants :

- Nature des entités manipulées : représentations numériques. Autrement dit, l'utilisateur manipule, non pas les entités de l'espace ambiant, mais des représentations numériques de ces entités. L'avantage est la souplesse apportée par le numérique (passage à l'échelle, situation de mobilité pouvant impliquer un objet hors de portée). L'inconvénient est d'insérer une indirection. Toutefois, les représentations numériques que nous proposons relèvent du vécu de l'utilisateur (elles ne sont pas arbitraires) : il s'agit de photos au format timbre poste des entités de l'espace ambiant. Ce choix est conforté par les résultats de notre méthode DisQo d'analyse des besoins chez l'habitant effectuée dans le cadre de ce projet (cf. article en Annexe 2).
- Type de manipulation : combinaison de manipulation directe et de paradigme langagier. L'expérience montre que la manipulation directe convient aux situations pour lesquelles l'utilisateur connaît le plan d'actions. Par exemple, l'utilisateur connaît la suite des actions à appliquer pour enregistrer un programme de Télévision (paradigme « du faire ») ; le paradigme langagier convient au cas où le plan d'actions est inconnu, mais où le but à atteindre l'est. Par exemple « enregistre le film de dimanche prochain sur TV5 » (paradigme du « faire faire »). Un focus group que nous avons réalisé l'an dernier sur la composition de services indique un grand intérêt pour l'utilisation de la langue naturelle. En conséquence, nous retenons le « principe d'égale opportunité » qui permet à l'utilisateur de choisir le paradigme qui lui convient et de passer de la manipulation directe au langagier de manière opportuniste. Cependant, la langue naturelle est ambiguë et il est difficile de fournir du « feedforward » c'est-à-dire, à partir d'un état donné, de rendre observables l'espace du possible. Pour contourner cette double difficulté, nous prenons comme inspiration, une « vieille » solution des années 80 à base de menus contextuels qui permet de construire des phrases correctes en langue (pseudo)naturelle [Tennant 83]. On trouvera en annexe 1 un descriptif de cette approche. CAMP s'est inspiré d'une approche semblable [Truong 04].
- Services offerts : tous + encapsulation + des aides au développement : mise au point, exécution « pour du beurre », retrouver des programmes.
- Niveau de contrôle : tous.
- Méta-IHM externe pour faciliter la réutilisation.

La figure 29 illustre les principes énoncés ci-dessus. L'ensemble des représentations (numériques) des entités manipulables est consigné dans une palette (à gauche). Pour permettre le passage à l'échelle (grandes quantités d'entités) l'affichage peut être conditionné par l'application de filtres (en haut à gauche). Exemples de filtre : toutes les entités (valeur par défaut), entités de la maison, de la cuisine, de la voiture, etc. Une seconde palette donne accès aux programmes ou couplages existants, elle aussi soumise à filtrage. Une troisième palette correspond aux outils de l'environnement de programmation (en haut, horizontalement).

L'environnement propose deux zones d'édition : édition en mode graphique (en haut à droite), édition en mode langage pseudonaturel dit mode LN (en bas). À tout instant, les contenus des zones d'édition en mode LN et en mode graphique sont sémantiquement équivalents : toute modification de l'une par l'utilisateur est automatiquement et instantanément reportée dans l'autre par le système (principe d'égale opportunité).

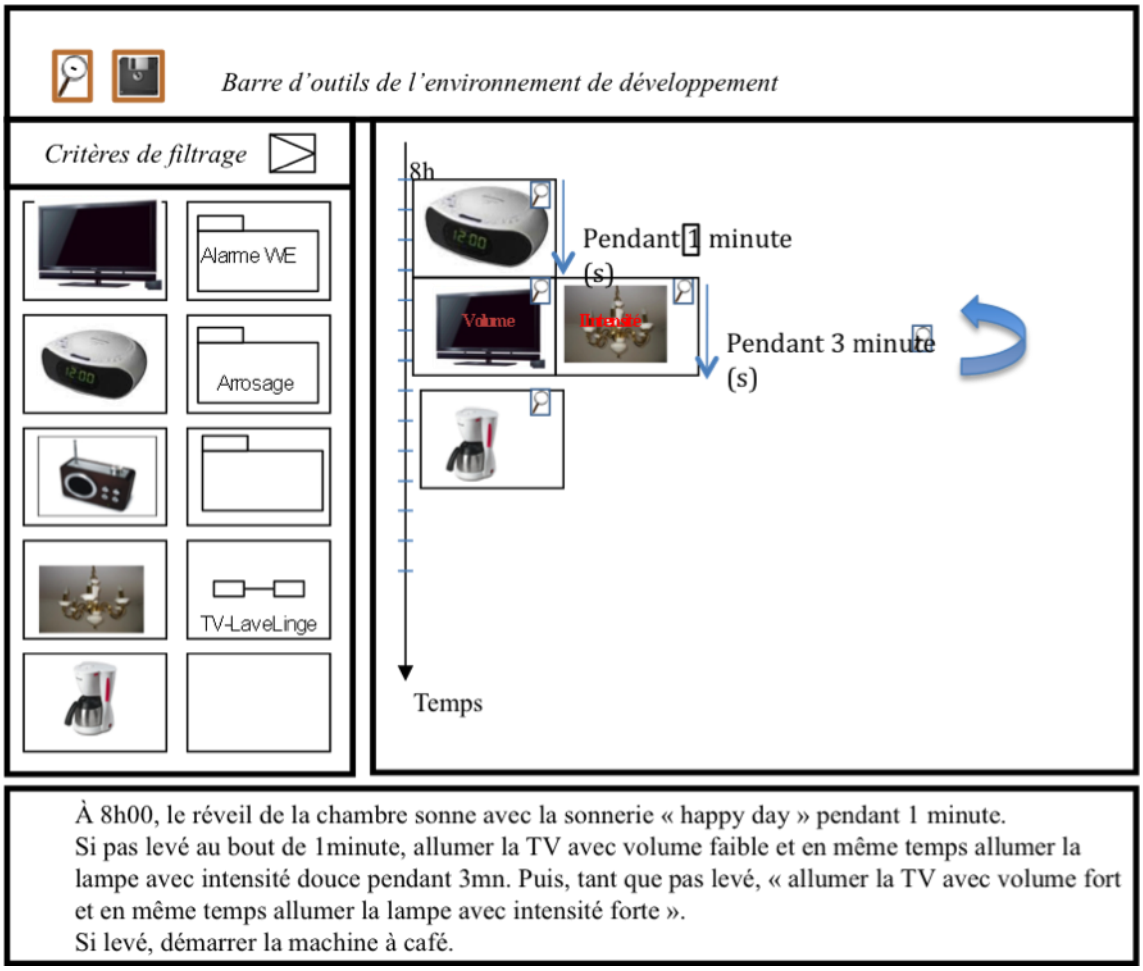


Figure 29. Vue générale de l'environnement de programmation de méta-IHM tactique.

6.3.2 Langage pseudonaturel

À l'heure de la rédaction de ce document, la grammaire du langage pseudonaturel n'est pas spécifiée. Sachant qu'une phrase de la méta-IHM doit être traduite dans le langage des AA (ou dans un langage pivot à projeter sur les AA), la sémantique du langage et sa syntaxe doivent être le résultat d'un compromis entre faisabilité-système et acceptabilité-utilisateur. Ce travail de réflexion sera engagé prochainement avec l'ensemble des partenaires académiques.

De la grammaire, dépendront les menus contextuels. L'annexe 1 donne une indication du style de l'interaction de construction de phrase en LN via des menus. Les menus linéaires proposés par Tennant et al. seront remplacés par des marking menus qui permettent de combiner les modes débutants et experts. Nous permettrons également la saisie directe au clavier (voire, le stylo Anoto sur papier si cette technique se révèle prometteuse) ou au moyen de représentants tangibles à l'instar des Media Cubes [Blackwell 04]. Le point clef sur le plan technique est de conserver la possibilité de projeter une même syntaxe abstraite en syntaxes concrètes distinctes. La figure 30 montre deux projections sous forme d'aimants numériques (pour CAMP [Truong 04]) ou de cubes (dans le projet AutoHan [Blackwell 04])

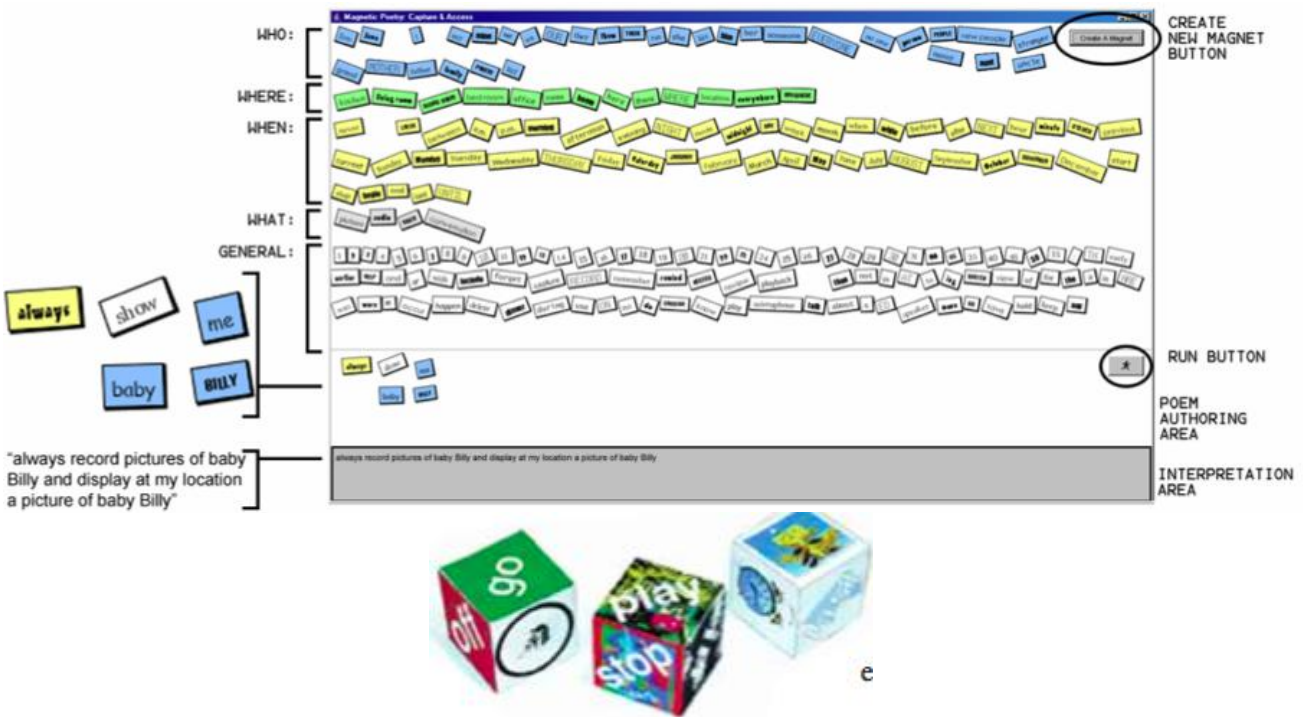


Figure 30. Deux syntaxes concrètes pour construire des phrases en langage pseudonaturel. En haut les « poetic magnets » de CAMP – ici des représentations numériques d'aimants porteurs d'un élément du vocabulaire ; en bas, les Media Cubes physiques de AutoHan porteurs de vocabulaire.

6.3.3 Langage graphique

Le vocabulaire du langage graphique correspond aux représentations timbre poste des entités manipulables. Nous distinguons deux types d'entités :

- Les entités mixtes de l'espace ambiant : TV, réveil, etc.
- Les entités numériques construites par l'utilisateur : les programmes (arrosage, alarme du Week-end, etc.) et les couplages réutilisables d'entités mixtes (par exemple, le couplage du téléviseur avec le lave-linge). L'existence de ces deux types d'entités numériques est motivée par les résultats de notre enquête (cf. Annexe 2).

La composition des éléments du vocabulaire s'appuie sur les opérateurs d'Allen (voir figure 31). Autrement dit, le temps est l'élément directeur de la construction de programmes en langage graphique. Dès lors, l'IHM de notre éditeur de programmes en langage graphique doit s'appuyer sur une métaphore temporelle. Nous avons retenu celle de l'agenda ou des programmes imprimés de télévision.

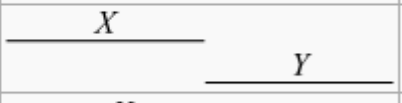
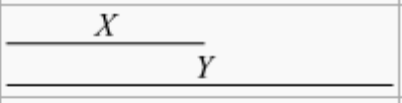
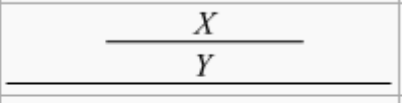
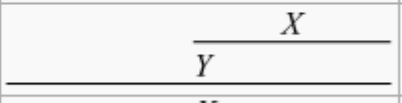
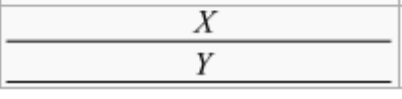
Relation	Illustration	Interpretation
$X<Y$ $Y>X$		X takes place before Y
$X\mathrm{m}Y$ $Y\mathrm{mi}X$		X meets Y (*i* stands for *inverse*)
$X\mathrm{o}Y$ $Y\mathrm{oi}X$		X overlaps with Y
$X\mathrm{s}Y$ $Y\mathrm{si}X$		X starts Y
$X\mathrm{d}Y$ $Y\mathrm{di}X$		X during Y
$X\mathrm{f}Y$ $Y\mathrm{fi}X$		X finishes Y
$X=Y$		X is equal to Y

Figure 31. Les opérateurs d'Allen entre intervalles.

Comme le montre la figure 29, à gauche de la zone d'édition, est affichée une ligne de temps verticale (celle-ci pourrait aussi être envisagée avec une présentation horizontale). L'utilisateur construit son programme par « glisser-déposer » d'éléments piochés dans les palettes d'entités. Dans l'exemple de la figure 29, l'utilisateur a placé le réveil à 8h du matin. Le réveil « meets » avec deux entités : la télévision et la lumière qui toutes deux fonctionneront sur le même intervalle de temps. « After » tout cela, la cafetière devra se mettre en route.

Si le temps constitue l'élément directeur de la construction de programmes, il ne suffit pas à spécifier le détail des comportements de chacune des entités. Un comportement d'entité se spécifie en « zoomant » sur sa représentation dans programme (double-clique ou sélection de la loupe visible sur la représentation timbre poste) : on entre dans une expression de type formulaire classique ou langage pseudonaturel. Les éléments essentiels du comportement sont observables (variables de paramétrage, conditions d'activation, conditions de fin, durée d'exécution, répétition). Dans l'exemple de la figure 29, le réveil va sonner pendant 3 minutes. Le volume du son de la TV et l'intensité lumineuse sont des variables paramètres. TV et lumière vont boucler.

6.3.4 Outils

Au moyen des outils de l'environnement de développement, l'utilisateur va pouvoir tester le programme en cours d'élaboration, et au final le sauvegarder (dans notre exemple, sous le nom « réveil-matin ») : c'est la possibilité d'encapsulation. Ce programme sera alors ajouté à la palette des entités numériques avec son nom en clair (au même titre qu' « Alarme » WE et « Arrosage »). Un outil indispensable est l'aide à la mise au point : « exécuter pour du beurre », montrer pourquoi « ça ne marche pas » (cf. Whyline [Myers 06]).

7 Conclusion

Ce document est la fusion des livrables D4.1 et D4.2 des sous tâches 4.1 et 4.2 telles que décrites dans le document de soumission. Avec les avancées du projet, nous avons jugé opportun de regrouper ces deux tâches et d'en définir les points d'ancrage avec l'architecture de CONTINUUM. L'avantage est d'intégrer explicitement l'utilisateur dans les trois niveaux d'adaptation envisagés. Les points de contrôle utilisateur sont aujourd'hui bien identifiés dans l'architecture CONTINUUM.

A contrario, comme le montre l'état de l'art et les nombreuses recherches actuelles sur le end-user programming, les pistes pour les méta-IHM sont nombreuses sans qu'aucune ne s'impose. La recherche sur ce point est très exploratoire. Il nous reste à maquetter nos premières propositions, à les évaluer avec des utilisateurs et réitérer avant de proposer une implémentation intégrée. Comme le laissent entendre Newman et Edwards avec leur notion de Template [Newman 09], nous pensons « viser juste » en permettant une vision centrée tâches et objectifs utilisateur : dans l'exemple ci-dessus, le programme « arrosage WE » aurait pu être élaboré par un spécialiste (l'installateur du système d'arrosage) ou par tout autre utilisateur final qui l'aurait rendu disponible sur le web (« social programming »). Notre utilisateur pourrait alors procéder par imitation assurant ainsi un « droit d'entrée » moins abrupte (pente douce).

8 Références bibliographiques

[Ayatsuka 05a] Ayatsuka, Y., Rekimoto, J., *tranSticks: Physically Manipulatable Virtual Connections*. In Proc. of CHI 2005, 251-260.

[Ballagas 03] Ballagas, R., Ringel, M., Stone, M., Brochers, J., *iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments*. In Proc. of CHI 2003, 537-544.

[Barralon 05] Barralon, N., Nguyen, V.T., Rey, G., *Techniques de couplage de bureaux : Ambient-Desktop comme illustration*. Conférence UBIMOB 2005, Deuxièmes Journées Francophones: Mobilité et Ubiquité 2005, ACM Press, 193-200.

[Baudisch 04] Baudisch, P., Xie, X., Wang, C., Ma, W.Y., *Collapse-to-zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content*. In Proc. of UIST 2004, Santa Fee,2004, 91-94.

[Beaudouin-Lafon 97] Beaudouin-Lafon, M., *Interaction instrumentale : de la manipulation directe à la réalité augmentée*. Dans les actes d'IHM'97 Cépaduès-Editions. 1997, 97-104.

[Bellotti 02] Bellotti V., Back M., Edwards K., Grinter R., Henderson A., Lopes C. *Making Sense of Sensing Systems: Five Questions for Designers and Researchers*. In Proc. of CHI 2002 Conf., Vol. 4(1), ACM New York, 2002, 415–422.

[Biehl 04] Biehl, J.T, Bailey, B.P, *ARIS: An interface for application Relocation in an interactive space*. In Proc. of GI 2004, Canadian Human-Computer Communications Society, 2004, 107-116.

[Blackwell 04] Blackwell, A.F. End-user developers at Home. Communication of the ACM, sept. 2004, Vol.47, no9, 2004, 65-66.

[Block 04] Block, F., Schmidt, A., Villar, N., Gellersen, H.W., *Towards a Playful User Interface for Home Entertainment Systems*. In Proc. of EUSAI'04, Eindhoven, Netherlands, 2004, 207-217.

[Bolin 05] Bolin, M. *End-user Programming for the Web.* MEng thesis, Massachusetts Institute of Technology, June 2005.

[Booth 02] Booth, K.S, Ficher, B.D, Lin, C.J.R, Argue, R., *The « mighty Mouse » Multi-Screen Collaboration Tool*. In Proc. of UIST 2002, 209-212.

[Brignull 04] Brignull, H., Lzadi,S., Fitzpatrick, G., Rogers, Y., Rodden, T., *The Introduction of a Shared Interactive Surface into a Communal Space*. Computer Supported Cooperative Work archive Proc. of the 2004 ACM conference, Illinois, 2004, 49 - 58.

[Calvary 01] Calvary, G., Coutaz, J., Thevenin, D. A *Unifying Reference Framework for the Development of Plastic User Interfaces*. In Proc. of EHCI'01, 173-192.

[Canfield Smith 82] Canfield Smith, D., Irby, C., Kimball, R., Verplank, B., Harslem, E., Designing the Star User Interface, 1982.

[Carroll 84] Carroll, J.M., Carrithers, C., *Training Wheels in a User Interface*. Communication of the ACM, vol. 27 (8), 1984, 800-807.

[Cooper 00] Cooper, S., Dann, W, Pausch, R., *Alice: a 3-D tool for introductory programming concepts*. Proc. of the fifth annual CCSC northeastern conference on The journal of computing in small colleges, Volume 15 Issue 5, 2000.

[Coutaz 05] Coutaz, J., Borkowski, S., Barralon, N., *Coupling Interaction Resources: an Analytical Model*, In Proc. of Eusai'05, 183-188.

[Coutrix 05] Coutrix, C., Nigay, L., Renevier, P, *Modèle d'Interaction Mixte : la Réalité Mixte à la lumière des Modalités d'Interaction*. Conférence UBIMOB 2005, Deuxièmes Journées Francophones: Mobilité et Ubiquité 2005, ACM Press, 153-160.

[Edwards 09] Edwards, K., Newman, M. Sedivy, J. and Smith, T. *Experiences with Recombinant Computing : Exploring Ad Hoc Interoperability in Evolving Digital Networks*. In ACM Transactions on Computer-Human Interaction, Vol. 16, No. 1, Article 3, April 2009.

[Fitzmaurice 95] Fitzmaurice, G., Ishii, H., Buxton, W., *Bricks: Laying the foundations for graspable user interfaces*. In Proc. of CHI'95, New York, 1995, 432-449.

[Cypher 91] Cypher, A., *EAGER: programming repetitive tasks by example*. In Proc. of the SIGCHI conference on Human factors in computing systems: Reaching through technology 1991.

[Cypher 93] Cypher, A., *Watch What I Do : Programming by demonstration*. MIT Press, Cambridge MA, 1993.

[Fishkin 04] Fishkin, K.P., *A taxonomy for and analysis of tangible interfaces*. Journal of Personal and Ubiquitous Computing, 2004, 8: 347–358.

[Fritz 91] Fritz, J.M., *HyperCard applications for teaching information systems*. In Proc of the twenty-second SIGCSE technical symposium on Computer science education SIGCSE '91, Volume 23 Issue 1.

[Girard 92] Girard, P., Environnement de programmation pour non programmeurs et paramétrage en conception assistée par ordinateur : Le système Like. Thèse HDR, Nov. 1992. Univ. Poitiers.

[Glinert 1987] Glinert E. *Out of Flatland: towards 3D Visual Programming*. IEEE FJCC'87, Dallas, Texas, 1987, 292-299.

[Gorbet 98] Gorbet, G.M., Orth, M., Ishii, H., Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. In Proc. of CHI'98, Los Angeles, 1998, 49-56.

[Gram 96] Gram, Ch., Cockton, G. (Eds.), *Design Principles for Interactive Software*. Chapman & Hall, London, 1996.

[Grolaux 05] Grolaux, D., Vanderdonckt, J., Van Roy, P., *Attach Me, Detach Me, Assemble Me Like You Work*. In proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'2005, Rome, 198-212.

[Han 00] Han, R., Perret, V., Naghshineh, M., *Websplitter : A unified XML Framework for Multi-Device Collaborative Web Browsing*. ACM Conference on Computer Supported Cooperative Work (CSCW) 2000.

[Harada 03] Harada, S., Hwang, J., Lee, B., Stone, M., *« Put-That-There »: What, Where, How? Integrating Speech and gesture in Interactive Workspace*. In International Conference on Computer Graphics and Interactive Techniques, 262–270.

[Hinckley 00a] Hinckley, K., Ramos, G., Guimbretiere, F., Baudish, P., Smith, M., *Stitching: Pen Gesture that Span multiple Displays*. UIST 2000, 91-100.

[Hinckley 00b] Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E., *Sensing Techniques for Mobile Interaction*. ACM UIST 2000 Symposium on User Interface Software & Technology, CHI Letters 2 (2), 91-100.

[Holmquist 99] Holmquist, L.E, Redström, J., Ljungstrand, P., *Token-Based Access to Digital Information*. Lecture Notes in Computer Science 1707, 1999, 234–245.

[Holmquist 01] Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H.W., *Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts*. In Proc. of Ubicomp 2001, Atlanta, Georgia, 2001, 116-221.

[Ingalls 88] Ingalls, D., Wallace, S., Chow, Y., Ludolph, F., Doyle, K., *Fabrik: a visual programming environment*. ACM SIGPLAN Notices, Conference proceedings on Object-oriented programming systems, languages and applications OOPSLA '88, 1988, Volume 23 Issue 11.

[Izadi 03] Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M., *Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media*. In Proc. Of UIST'03, Vancouver, Canada, 2003, 159-168.

[Johanson 02] Johanson, B., Hutchins, G., Winograd, T., *PointRight: Experience with Flexible Input Redirection in Interactive Workspaces*. In Proc. of UIST-2002, 227 - 234 .

[Lepreux 06] Lepreux, S., Vanderdonckt, J., TowardS a Support of the user interfaces design using composition rules. CADUI 2006, à paraitre.

[Lieberman 06] Lieberman, H., Paternò, F. and Wulf, V. *End-User Development*. Human-Computer Interactions Series, Vol. 9, Springer, 2006.

[Lieberman 2001] Lierberman, H., *Your Wish Is My Command : Programming by Exemple*. MK Publishers Media Lab, Massachusetts Institute of Technology, 2001.

[Mackay 90] Mackey, W. *Users and Customizable Software : a co-adaptive phenomenon*. PhD Thesis, Sloan School of Management, MIT, 1990.

[Marcopoulos 04] Markopoulos, P., Mavrommati, I., Kameas, A., *End-User Configuration of Ambient Intelligent Environments: Feasibility from a User Perspective*. In Proc. of EUSAI'04, Eindhoven, Netherlands, 2004, 243-254.

[McNerney 04] McNerney, T.S, *From turtles to Tangible Programming Bricks: explorations in physical language design*. Journal of Personal and Ubiquitous Computing, 2004, 8: 326-337.

[Maulsby 89] Maulsby, D.L., Witten, I., Kittlitz, K.A, *Metamouse: specifying graphical procedures by example*. In Proc. of the 16th annual conference on Computer graphics and interactive techniques SIGGRAPH '89, 1989, Volume 23 Issue 3.

[Myers 86] Myers, B, Buxton, W., *Creating highly-interactive and graphical user interfaces by demonstration*. ACM SIGGRAPH Computer Graphics, Proc. of the 13th annual conference on Computer graphics and interactive techniques SIGGRAPH '86, 1986, Volume 20 Issue 4.

[Myers 06] Myers, B., Weitzman, D.A., Ko, A.J., Chau, D.H, *Answering Why and Why Not Questions in User Interfaces. In Proc. of* CHI 2006, Understanding Programs & Interfaces, 2006, Montréal.

[Myers 00] Myers, B., Hudson, S.E, Pausch, R., *Past, present, and future of user interface software tools*. ACM Transactions on Computer-Human Interaction (TOCHI), 2000, Volume 7 Issue 1.

[Myers 02] Myers, B., *Mobile Devices for Control*. The Fourth Symposium on Human-Computer Interaction for Mobile Devices, Mobile HCI'02, Pisa, Italy, Sept 18-20, 2002, 1-8.

[Nardi 95] Nardi, B., A. A Small Matter of Programming – Perspectives on end user computing. The MIT Press, Cmabridge, Massachussets, 1995 (second edition).

[Newman 02] Newman, M.W., Sedivy, J.Z., Neuwirth, C.M., Edwards, W.K., Hong, J.I., Izadi, S., Marcelo, K., Smith, T.F, *Designing for Serendipity: Supporting End-User Configuration of Ubiquitous Computing Environments*. In Proc. of Designing Interactive Systems (DIS), London 2002, 147-156.

[Newman 08] Newman, M., Elliott, A., and Smith, T.F. *Providing an integrated user experience of networked media, devices, and services through end-user composition*. In Proc. Of the International Conf. On Pervasive Computing (Pervasise'080, 2008.

[Norman 86] Norman, D.A., Draper, S.W., *User Centered System Design: New Perspectives on Human-Computer Interaction*. Erlbaum Associates, Hillsdale, NJ, 1986.

[Norman 99] Norman, D.A., *Affordance, Conventions, and Design. Interaction*s. CACM Publ., 1999, 38-42.

[Pane 06] Pane, J. Myers, B., *Chapter 3 : More natural programming languages and environments. In End-User Development*. Computer Interactions Series, Vol. 9, Springer, 2006, 31-50.

[Papert 80] Papert, S., *Mindstorms : Children, Computers, and Powerful Ideas*. New York : Basic Books, 1980.

[Pering 05] Pering, T., Ballagas, R., Want, R., *Spontaneous marriages of Mobile Devices and Interactive Spaces*. Communications of the ACM, Volume 48, Issue 9, 2005, 53 – 59.

[Ponnekanti 01] Ponnekanti, S.R., Lee, B., Fox, A., Hanrahan, P., Winograd, T., *ICrafter: A Service Framework for Ubiquitous Computing Environments*. In Proc of Ubiquitous Computing Conference. Lecture Notes in Computer Science", 2001, 56-75.

[Pinna-Dery 04] Pinna-Dery, A.M , Fierstone, J., Riveill, M., Picard.E., User Interface: a Technical Component in Component-Based Models in Response to Human Computer Interaction Adaptation. Rapport de recherche, février 2004.

[Rekimoto 98] Rekimoto, J., *A multiple-device approach for supporting whiteboard-based interaction*s. In Proc. of ACM CHI'98, 1998.

[Rekimoto 03] Rekimoto, J., Ayatsuka, Y., Kohno, M., *SyncTap: An Interaction Technique for Mobile Networking*. In Proc of MOBILE HCI'03, Udine, Italy, 2003, 104-115.

[Rekimoto 01] Rekimoto, J., Ullmer, B., Oba, H., *DataTiles: A Modular Platform for Mixed Physical and Graphical Interactions*. In Proceedings of CHI'01, Seattle, Washington, 2001, 269-276.

[Repenning 04] Repenning, A., & Ioannidou, A, *Agent-Based End-User Development*. Paper in the Communications of the ACM, September 2004, Volume 47, Number 9, 43-46.

[Resendez 03] Resendez, K., Bachnak, R., *LabVIEW programming for internet-based measurements*. Journal of Computing Sciences in Colleges archive,Volume 18 , 2003 79 – 85. [Rodden 04] Rodden, T., Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Akesson, K.P., Hansson, P., *Configuring the Ubiquitous Home*. In Proc of the 2004 ACM Symposium on Designing Interactive Systems, August 1st-4th, Cambridge, Massachusetts: ACM Press.

[Resnik 09] Resnik, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. Scratch : Programming for all. Communication of the ACM, Nov. 2009, Vol. 52, No11, 2009, 60-67.

[Shneiderman 87] Shneiderman, B., Designing the User Interface, Strategies for Effective Human-Computer Interaction. Addison Wesley Publ., 1987.

[Smith 94] Smith, D.C., Cypher, A. & Spohrer, J., *KidSim: Programming Agents Without a Programming Language*. In Communications of the ACM, 37(7), July 1994, 54 - 67.

[Sohn 06] Sohn, T.Y., Dey, A.K., *iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications*. In Proc. of the International Conference on Pervasive Computing 2006. Dublin, Ireland, May 2006, 974–975.

[Solomon 78] Solomon, C.J., *Teaching young children to program in a LOGO turtle computer culture*. ACM SIGCUE Outlook, 1978, Volume 12 Issue 3 July.

[Sousa 03] Sousa, J.P., Garlan, D., *The Aura Software Architecture: an Infrastructure for Ubiquitous Computing*. Carnegie Mellon Technical Report, CMU-CS-03-183, August 2003.

[Stewart 98] Stewart, E.M., Raybourn, B., Bederson, A. Druin, *When Two Hands Are Better Than One: Enhancing Collaboration Using Single Display Groupeware*. In CHI '98: CHI 98 conference summary on Human factors in computing systems, 287-288, Los Angeles, ACM Press.

[Streitz 99] Streitz, N.A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C, Reischl, W., Rexroth, P, Seitz, P., Steinmetz, R., *i-LAND: an interactive landscape for creativity and innovation*. In Proc of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, 1999.

[Tandler 01] Tandler, P., *The BEACH Application Model and Software Framework for Synchronous Collaboration in Ubiquitous Computing Environments*. In Proc of the 3rd international conference on Ubiquitous Computing table of contents, Atlanta, 2001, 96 – 115.

[Tennant 83] Tennant, H.R., Ross, K. and Thompson, C. Usable natural language interfaces through menu-based natural language understanding. In Proc. Computer Human Interaction, CHI'83, ACM Publ., 1983, 154-160.

[Truong 04] Truong, K.N., Huang, E.M, and Abowd, G. *CAMP : a magnetic poetry interface for end-user programming of capture applications for the home*. In Proc. Of the 6th international conf. On ubiquitous computing (Ubicomp'04), 2004, 143-160.

[Ullmer 98] Ullmer, B., Ishii, H., Glas, D., *MediaBlocks: Physical Containers, Transports, and Controls for Online Media*. In Proc of SIGGRAPH'98, Orlando, 1998, ACM Press, 379-386.

[Vanderdonckt 01] Vanderdonckt, J., Bouillon, L., Souchon, N., *Flexible Reverse Engineering of Web Pages with VAQUITA*. In Proc of IEEE 8th Working Conference on Reverse Engineering WCRE'2001 (Stuttgart, 2-5 octobre 2001, IEEE Press, Los Alamitos, 2001, 241-248.

[Wallace 04] Wallace, G, Bi, P., Li, K., Anshus, O., *A Multi-Cursor X Window Manager Supporting Control Room Collaboration*. Princeton University, Computer Science, Technical Report TR-707-05, July 2004.

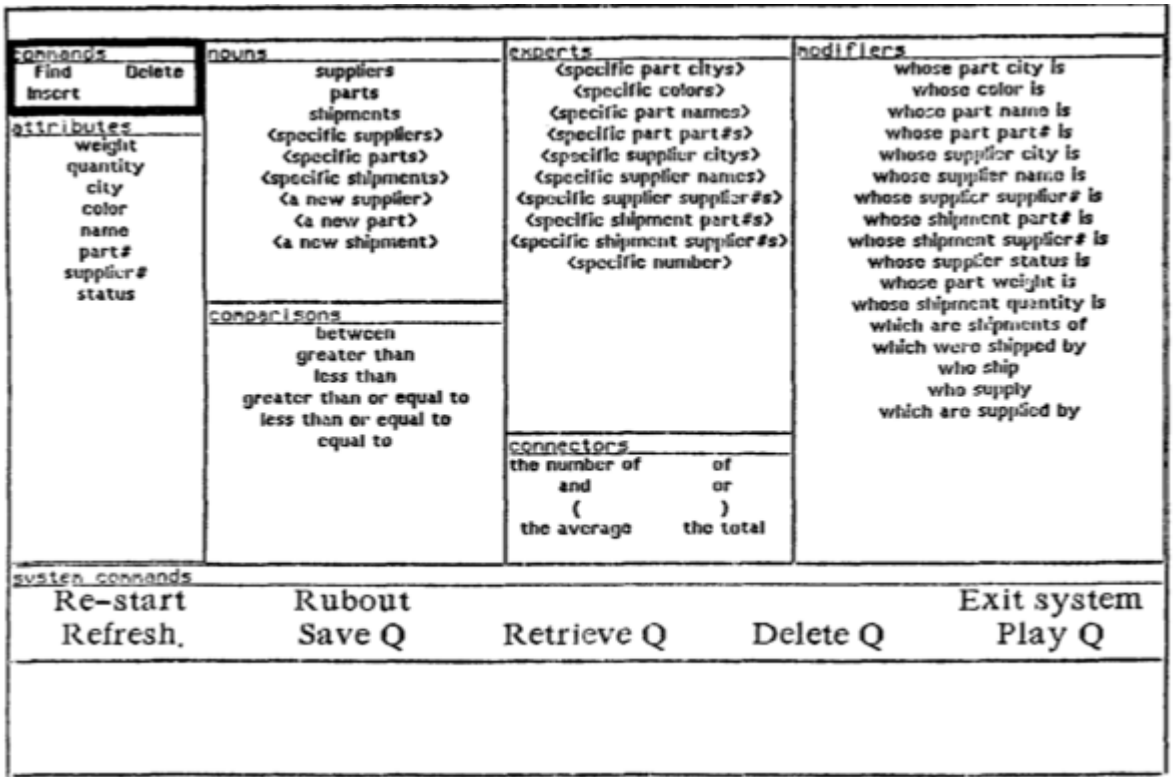
[Weiser 91] Weiser, M., *The Computer for the Twenty-First Century*. Scientific American, September 1991, 94-10.

[Whitley 97] Whitley, K.N., *Visual programming languages and the empirical evidence for and against*. Journal of Visual Languages and Computing 8(1), 9-142.

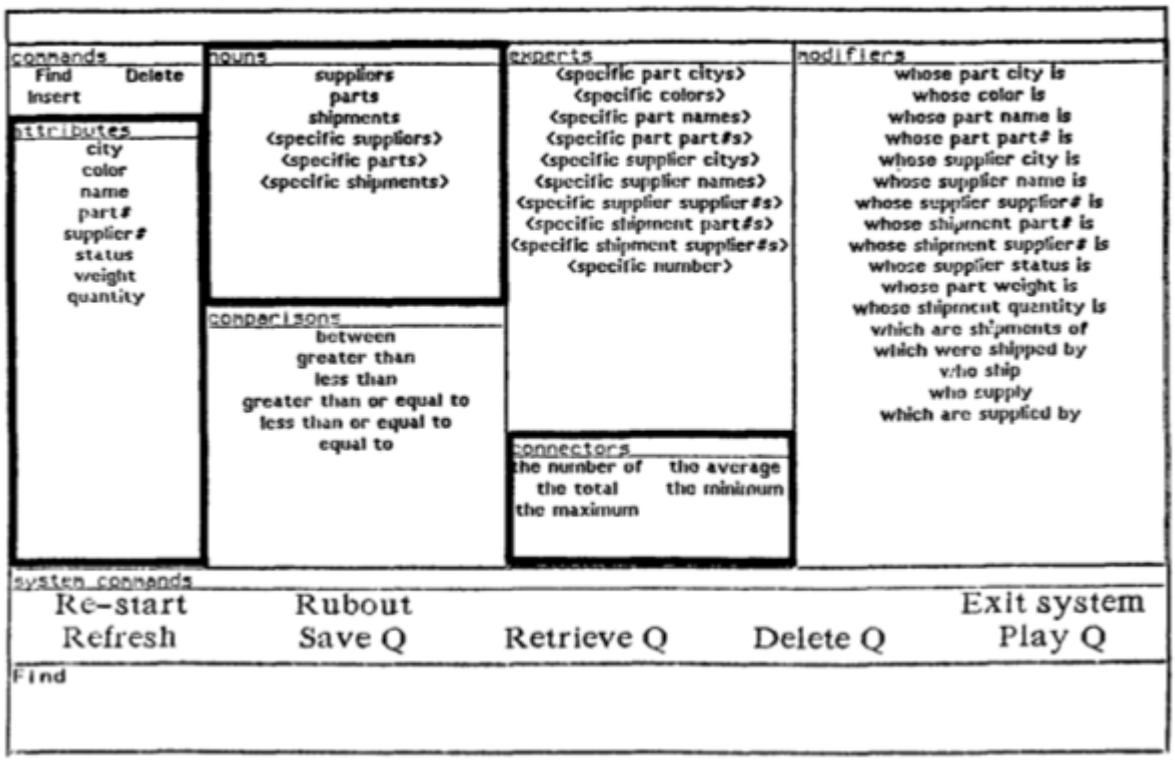
9 Annexe 1 – Menus contextuels et langue naturelle

Exemple tiré de [Tennant 83] montrant comment s'effectue la construction de phrase correcte en langue pseudo-naturelle au moyen de menus contextuels. Les menus syntaxiquement valides s'allument au fur et à mesure de la construction de la phrase Cette technique assure à la fois le feedback et le feedforward : l'utilisateur peut évaluer instantanément l'effet de ses actions tout en prenant connaissance de l'ensemble des actions possibles.

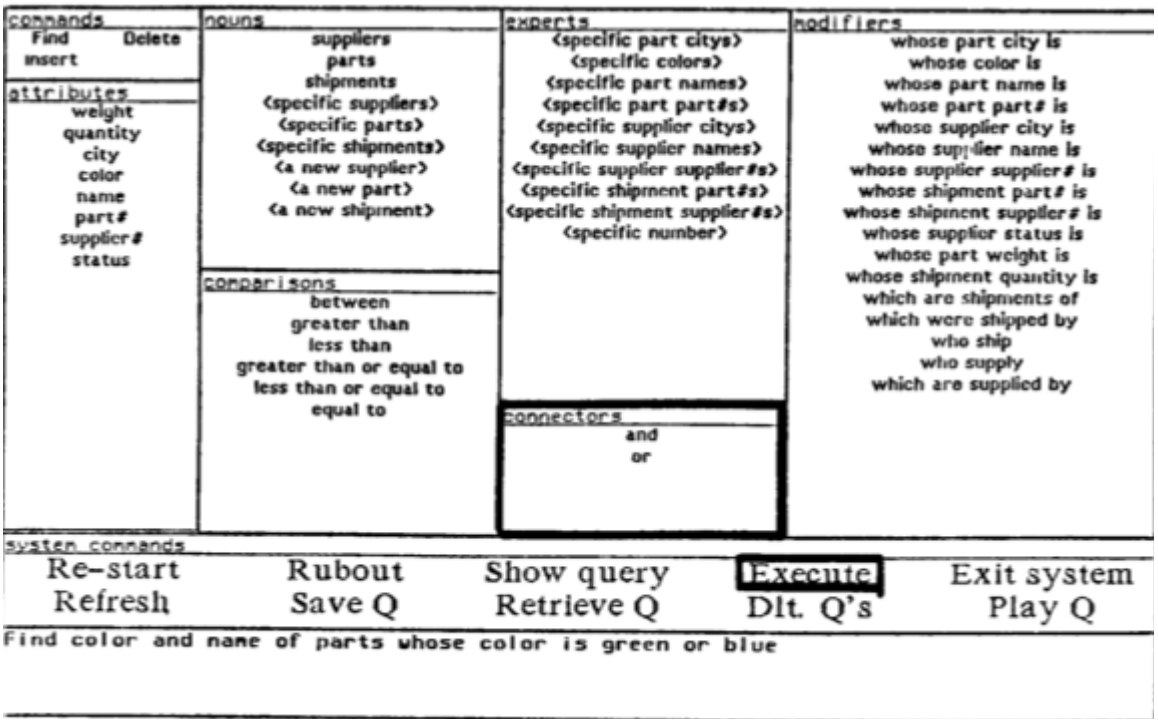
Dans cet exemple, l'utilisateur doit commencer par un nom de commande (find, insert, delete) :



L'utilisateur a choisi « Find » : le verbe correspondant est affiché dans la zone de construction de la phrase (en bas) et les menus valides s'allument (attributes, nouns, connectors) :



L'utilisateur choisit « color », « and », « name », « of », « parts » et « whose color is », etc. par une succession de menus contextuels pour aboutir à une commande complète qui peut alors être exécutée :



10 Annexe 2 – Méthode DisQo et résultats

On trouvera ci-dessous l'article que nous avons présenté au workshop « End-User Programming for Services » tenu à Rome en juin 2010 en liaison avec la conférence AVI 2010.

About Composing Our Own Smart Home

Joëlle Coutaz, Emeric Fontaine
Grenoble University
Grenoble Informatics Laboratory
BP 53, 38041 Grenoble Cedex 9
+33 4 76 51 48 54
{joelle.coutaz,emeric.fontaine}@imag.fr

Nadine Mandran
CNRS
Grenoble Informatics Laboratory
BP 53, 38041 Grenoble Cedex 9
+33 4 76 57 48 96
nadine.mandran@imag.fr

Alexandre Demeure
Grenoble University, INRIA
655 Avenue de l'Europe
38330 Montbonnot-Saint-Martin
+33 4 76 61 54 71
alexandre.demeure@inrialpes.fr

ABSTRACT

This paper reports on an empirical study designed as a follow-up of a theoretical model intended to support reasoning about the composition of smart artifacts by end users. We have solicited 17 families and used a combination of interviews and playful cultural probes. Results show that families are willing to couple smart objects to improve their lives, and that the theoretical questions raised by our model are sound.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User interfaces.*

General Terms

Design, Experimentation, Human Factors.

Keywords

End-User composition, smart artifacts coupling, smart home, ubiquitous computing, service-oriented computing.

INTRODUCTION

Ubiquitous computing promises unprecedented empowerment from the flexible and robust combination of software services with the physical world. Software researchers assimilate this promise as system autonomy where users are "conveniently" kept out of the loop. Their hypothesis is that services, such as music playback and calendars, are developed by service providers and pre-assembled by software designers to form new service front-ends. Their scientific challenge is then to develop secure, multi-scale, multi-layered, virtualized infrastructures that guarantee service front-end continuity. Although service continuity is desirable in many circumstances, end users, with this interpretation of ubiquitous computing, are doomed to behave as mere consumers, just like with conventional desktop computing.

Another interpretation of the promises of ubiquitous computing, is the empowerment of end users with tools that allow them to create and reshape their own interactive spaces. The mashup paradigm incarnates this view for networked knowledge and services. Mashups, however, are concerned with the digital dimension of our world, not with the combination of the digital with the physical. In this paper, we focus on the composition of smart artifacts by end users, bringing together physicality and digital power. Examples of such artifacts include smart phones, augmented fridges, or information appliances.

Our hypothesis is that end users are willing to shape their own interactive spaces by coupling

AVI'10, May 25-29, 2010, Rome, Italy.

smart artifacts, building imaginative new functionalities that were not anticipated by system designers. A number of tools and techniques have been developed to support this view including the Jigsaw editor [5], CAMP [8], iCAP [4], or Newman's work on end user composition with OSCAR [6]. The major focus of this prior work is on exploring novel interaction techniques and on technical frameworks. In this paper, we are concerned with the fundamental meaning (and human needs) of building confederation of interoperating smart artifacts.

In [2], we present a chemistry-inspired theoretical model that supports reasoning about the composition of smart artifacts by end users. Whereas scientific knowledge in chemistry is sufficiently advanced to predict the occurrence and results of a reaction, such knowledge is clearly lacking in ubiquitous computing. In particular, we are unable to predict which artifacts of every day life end users would be willing to couple (and decouple) to obtain new services. Is coupling commutative, associative, distributive over some other operation? In other word, can we define an algebra over smart artifacts so that we can generalize the problem and reason at a high level of abstraction in a rigorous manner?

The field study presented in this paper is our first attempt at answering the questions raised by our theoretical model. The experimental study is presented in the next section. Our findings are discussed in the last section.

THE EXPERIMENTAL STUDY

Participants

Drawing on Davidoff's et al. experiment and conclusions (i.e. "families want more control of their lives" [4]), we focused on "busy" families. The participants have been solicited through bulletin board advertisements, email, as well as from personal relationships.

We have recruited 17 families representing a total of 40 persons (35 adults and 5 children), all living in the area of Grenoble (France). Of the 17 families, 12 were dual income families, 1 was single parent, 2 were house mates, and 2 were retired couples. All families were well educated with medium to high standard of living.

Method

Our method was designed to reconcile the following requirements: to collect meaningful data in a minimum of time while respecting privacy. As presented above, we are interested in determining how far people are ready to envision the interconnection of everyday devices to improve control of their lives, and to which extent coupling objects is commutative, associative and distributive. For so doing, we have used a combination of interview [7] (good for clarification), playful cultural probe (appropriate for respecting privacy and for improving subjects involvement [1]). The presence of the experimental team (ourselves, from 1 to 3 persons) was limited to 1h30 per family home. Fieldwork was structured as a four-step process: photographing, interview, game, and debriefing.

Step 1: Photographing. Using digital cameras provided by the experimental team, two volunteer family members were asked to take pictures of 10 objects at the rate of 2 objects per room. For each of the 5 rooms of their choice, they were asked to take the picture of one object that they considered to be necessary in their every day life or that would help them in organizing their lives, as well as the picture of one object that they considered to be superfluous but valuable (typically, a painting). The volunteers (in general, the parents) were not supposed to be in the same room at the same time so that they would not know which pictures the other member had taken. Meanwhile, the experimental team would wait sitting at a place indicated by the parents (typically, the living room where they usually meet with friends and visitors).

Step 2: Interview. We then conducted an interview with all the family members, using the pictures as input material. Questions were directed at understanding the reasons for their choices, the value attached to the objects or the services provided in daily use. Special attention was given to

the (many) remote controller(s) typically found in the household environment. We progressively oriented our questions towards novel uses of smart artifacts. In particular, we asked which objects of the house (including those on the pictures) they would qualify as "programmable" (e.g., TV's, washing machines, alarm clocks), "communicating" (e.g., computers, mobile phones), or emotional (i.e. carrying intimate value). This was used as a means to elicit routines and exceptional needs as well as to prepare the game developed in Step 3.

Step 3: Association game. The association game drew on people creativity using the pictures as play cards. Pictures were sorted randomly and presented two at a time (then, three at a time) on a tablet PC. Family members were asked to imagine which service(s) and value(s) these two (or three) objects coupled together would provide them with. Random coupling was designed to solicit imagination in unexpected ways, and to get hints about the existence of a "natural" algebra over smart artifacts.

Step 4. Debriefing and informal discussion. The last stage was dedicated to debriefing, including opened friendly discussions.

Overall, we have collected comments and objective data for 349 couplings for a total duration of 25 hours of our presence in the 17 family homes.

Data Analysis

Interviews and debriefings helped us to identify recurring facts between home families such as key moments during weekdays for which families would expect support from a smart home, or attitudes with regard to "programming the home". Data from the association game as well as from the interviews were used to find answers to our theoretical questions. More specifically, we classified the objects that have been photographed into four categories: objects that have been denoted as "programmable" by the subjects, objects that have been declared as "communicating", objects that support both capabilities, and objects that have none of these two properties. Using the Chi-square test, we have been able to find strong significance between the abilities of the subjects to envision (or not) services depending on the capabilities of the assembled objects. In particular, the communication capability allows peoples to more easily imagine new services from the assembled object.

FINDINGS

Our experiment has led to three types of results: recurring facts across families, early answers to our theoretical questions, as well as insights about our method.

Recurring Facts

We found a number of facts that are quite consistent with the results reported in prior literature:

1. "Wake-up" time, "on-the-way-to-home" and "arriving-home" times are key to people. To save time and improve efficiency, activities are organized into well-polished procedures. As a result, exceptions to these routine tasks are sources of stress. Support for avoiding or for solving exceptions is one class of services expected from a smart home. This includes the management of possessions (laundry to be launched because of a business trip planned in a couple of days, food on the point to be missing, medicine close to expiration date), decision-making (what to buy, what to wear today), reminders (doctor appointment), security (door properly locked), resources consumption and resource sharing among family members (typically, hot water and bathroom occupation in the morning), etc.
2. With regard to programming, attitudes range from "I do not want to be assisted" to "It will work 99% of the time, but it will be hell for the other 1%". Motivation for programming is systematically grounded on a clear straight forward observable benefit.

We believe that our findings related to coupling everyday life objects are original.

About Coupling

Our data from the association game shows two important results: (1) Family members are prone to envision new services when coupling involves one "communicating" object, or one "programmable" object, at least. (2) The "communicating" capability has more impact than "programmability" on the

capacity of family members to imagine new services. However, 78 of the 349 couplings resulted in service finding although none of the objects were programmable or communicating. For example, the couple "bed-shower", whose objects had not been classified as programmable nor communicating, suggested that "getting up from the bed in the morning would turn the shower on in order to provide water at the right temperature when coming back from the toilet". This means that there is a large body of potentiality for novel services based on mundane everyday objects[7].

The services suggested by our family members fall into four categories. We illustrate them with the most typical examples drawn from our fieldwork.

Service substitution. People have observed that, for the same (sport) events, commentaries on radio broadcasts are richer than those provided by TV. As a result, they would like to replace the TV sound service with that of the radio to improve the overall quality of the informational experience. Another example is User Interface substitution: some people are quite skill at setting up alarms on their mobile phone, but they do not know how to do this for their physical home alarm clock. As a consequence, they would find it quite convenient to replace the user interface of their alarm clock with that of their phone thanks to a convenient opportunistic coupling of the phone with the alarm clock.

Service improvement. Some household appliances such as washing machines and cubboards, do not provide any convenient way to control and monitor their current internal state. Appliances than are not sufficiently equipped could be improved by coupling them with additional input and output facilities such as those of the TV set.

Service chaining. Service chaining is intended to improve comfort, wellbeing as well as resources for the routine, but hectic, activities. For example, "picking up the towel after the shower would trigger the coffee machine so that coffee would be ready just in time, at the right temperature, along with the radio turned on in the kitchen broadcasting the news using the appropriate sound level".

Service "starter". We have observed that some appliances serve as triggers for services that are expected to be pre-composed to support routine activities. The towel and the bed mentioned above, play this role, implicitly. Not surprisingly, people also want to have an explicit and reliable control over the home (cf. the worry that 1% of the time, the house would turn into hell). Some people came up with the "*morning starter push button*" conveniently located close to the bed that would gently "wake up" the house when pushed.

The need for chains of services underpins some form of associativity. For example, one family qualified the "towel-coffee machine" coupling as a "morning package". If the expression *towel – coffee* denotes the coupling of the towel with the coffee machine, then (*towell – coffee)*, between parenthesis, denotes the notion of package. During the discussion, our family members thought of adding the "morning starter push button" *b* to get a controllable chain "*b – t – c*". Here, their mental construction can be formalized as a right associativity of the "coupling" operation: *b – t – c = b – (t – c)*.

Coupling for service improvement entails some form of distributivity. Typically, the TV set *tv* has often be mentioned as a way to observe and control the state of a number of appliances such as the washing machine *w* or the oven *o*. This can be formalized in the following way: *tv – (w | o) = tv – w | tv – o*. Commutativity is generally satisfied with notable exceptions when there is a causality relationships between the objects.

About the Experimental Method

The "Snapshots taking" of our fieldwork has multiple advantages: (1) It serves as an ice breaking between the family members and the experimental team; (2) Family members "reveal their house"

[7] Detailed quantitative data will be presented in a full paper.

naturally while we, the experimenters, do not intrude their private spots. (3) Family members get truly involved (and intrigued by what will come next). (4) As opposed to playful probing proposed by R. Berhaupt [1], our game uses images of intimate objects, not of generic entities. This increases the interest and imagination of the participants while improving the meaningfulness of the data collected.

CONCLUSION

A central focus of our work is investigating the fundamental meaning of building confederation of interoperating smart artifacts. Our approach to this problem is theory-driven with the quest for an algebra that would support generalization and prediction. Although additional investigations are necessary, early results from our fieldwork support this approach.

ACKNOWLEDGMENTS

This work has been supported by the ANR CONTINUUM project reference ANR-08-VERS-005.

REFERENCES

[1] Bernhaupt, R., Weiss, A., Obrist, M. & Tscheligi, M. (2007) Playful Probing: Making Probing more Fun. In *INTERACT 2007*, Springer LNCS, 606-619.

[2] Coutaz, J. (2008) End-User Programming and the Intrinsic Complexity of Networked Artefacts. In *4th Workshop on End-User Software Engineering*, ICSE 2008.

[3] Davidoff, S., Lee, M.K., Yiu, C. Zimmerman, J. & Dey, A. (2006) Principles of Smart Home Control. In *Ubicomp 2006*, LNCS 4206, Springer Verlag Berlin Heidelberg, 19-34.

[4] Dey, A., Sohn, T., Streng, S. & Kodama, J. (2006) iCAP: Interactive prototyping of context-aware applications. In Pervasive 2006, Springer, 254-271.

[5] Humble, J., Crabtree, A., Hemmings, T., Akesson, K.P., Koleva, B., Rodden, T. & Hansson, P. (2003) "Playing with the bits" user-configuration of ubiquitous domestic environments. In *Ubicomp 2003*, 256–263.

[6] Newman, M.W., Elliott, A. & Smith, T.F. (2008) Providing an integrated user experience of networked media, devices, and services through end-user composition. In *Pervasive 2008*, Springer Verlag, 213–227.

[7] Silverman, D. (1997). Introducing qualitative research. In D. Silverman (Ed.), Qualitative research. Theory, method and practice London:Sage, 1-7.

[8] Truong, K.N., Huang, E.M. & Abowd, G. (2004) CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In Ubicomp 2004, Springer, 143-1